

Prepared By:
Amr Swafta

ancopra@yahoo.com

أولا:العناصر التي تتكون منها لغة (C++) :

1) الرموز الأساسية:

- جميع الحروف اللغة الإنجليزية (كبيرة أو صغيرة)

A,B,C,.......a,b,c......

- جميع أرقام اللغة الإنجليزية:

0,1,2,3,4,5,6,7,8,9

- الرموز الخاصة:

قد تتساءل ما هي الرموز الخاصة؟أجيبك بكل بساطه هي كل رمز قد يخطر في بالك وهذه مجموعة رموز خاصة لكي تتعرف عليها

! ,) , - , + , = , * , ^ , % , $, # , } , * ,.....................

2) الكلمات الأساسية (الكلمات المجوزة)

وهي مجموعة من الكلمات تستخدمها لغة (C++) لأداء وظائف محدده ولكن عليك الانتباه فهذه الكلمات لا تستخدم إلا فقط للوظيفة المخصصة لها فقط وهذه مجموعة من الكلمات المحجوزة

for , if , else , case , int , cout , cin , void ,.......

- ملاحظات على الكلمات المحجوزة:

1) تظهر الكلمات المحجوزة بلون ازرق عند كتابتها

2) لا يجوز أن تستخدمها إلا لوظيفتها المخصصة(فلا يجوز أن تكون أسماء لمتغيرات)

ثانيا:المتغيرات :

وهي مواقع في الذاكرة يستخدمها المبرمج لكي يخزن بها قيم معينة وتتم عملية إعطاء المتغيرات قيم بطريقتين:

الأولى: إعطاء المتغير قيمة ابتدائية أثناء كتابة البرنامج (بواسطة جملة التعيين)

مثال :

```
int x=5;
```

الثانية: إعطاء المتغير قيمة أثناء تنفيذ البرنامج (بواسطة جملة الإدخال)

مثال :

```
int x;
cin>>x;
```

- هناك شروط يجب مراعاتها عند تسمية المتغيرات وهي :

1) يجب أن يبدأ اسم المتغير بحرف

2) لا يحتوي اسم المتغير على فراغ

3) أن لا يكون اسم المتغير هو احد الكلمات المحجوزة

4) لا يحتوي اسم المتغير على احد الرموز الخاصة

- يعتبر اسم المتغير حالة حساسة (كيف؟؟؟)

أي أن الحرف الكبير يختلف عن الحرف الصغير في اسم المتغير فمثلا :

Avg , AVG , avg , aVg جميعها أسماء متغيرات مختلفة

- هناك رمز خاص وحيد يمكن استخدامه في أسماء المتغيرات وهو (_)(underscore)

- أمثلة على أسماء متغيرات صحيحة :

A , sum , y2 , h , H , name , max , sub , avg , x_y , sum_number90

- أمثلة على أسماء متغيرات خاطئة :

2name ← ابتدأ برقم
a.b ← احتوى على رمز خاص
su m ← احتوى على فراغ
Rf#u ← احتوى على رمز خاص

- لكل متغير هناك نوع للبيانات التي سوف تخزن به ومن أنواع البيانات الأكثر استخداما :

int مثل: (0 , 70 , 999 ,-9 , -54)
float مثل: (4.0 , -5.0 , 0.0 ,-0.7 , 0.1)
double مثل: (55.98 , 13.15 , -77.987 , 13.4)
char مثل: (a , A , b , c , O) يجب أن يكون حرف واحد فقط وبين إشارتي اقتباس مفردة
bool وتأخذ فقط قيمتان هما true وfalse

ثالثا: الثوابت :

- هي مواقع في الذاكرة يتم حجزها من قبل المبرمج لتخزين قيم فيها ولكن هذه القيم تبقى ثابتة طوال فترة تنفيذ البرنامج بعكس المتغيرات التي يمكننا تغير قيمتها.

- تعرفنا على المتغيرات والثوابت وأنواع البيانات وألان سوف نتعرف على كيفية استخدام هذه المتغيرات والثوابت:

- <u>كيف استخدم المتغيرات ؟؟؟</u>

عندما تريد أن تستخدم متغير في برنامجك فيجب عليك أن تعرف المتغير ويمكنك إعطاءه قيمة ابتدائية إذا أردت وذلك بالطريقة التالية:

DataType Variable_name = value ;

هذا يعني أننا نقوم بتحديد نوع البيانات التي نريد تخزينها في المتغير ثم نكتب اسم المتغير حسب الشروط التي تعرفنا عليها سابقا وإذا أردنا إعطاءه قيمة ابتدائية نضع الرمز الخاص وهو المساواة ثم القيمة المراد تخزينها بالمتغير ثم ننهي هذه الجملة بفاصلة منقوطة(;).

- أمثلة على تعريف المتغيرات :

int a;
double dd;
float sum = 0.8;
double H_h = 77.90;
char C = 'a'; (عندما يكون نوع البيانات المراد تخزينها بالمتغير من نوع char يجب أن تكون قيمة المتغير حرف واحد بين إشارتي اقتباس فردية)
int x,y=9; (تم تعريف متغيرين وإعطاء المتغير y قيمة أولية وهي الرقم 9 ولم يعطى المتغير x أي قيمة أولية)
float sum,avg; (تم تعريف متغيرين من دون أن نعطي المتغيرات قيم ابتدائية)

- <u>كيف استخدم الثوابت؟؟؟</u>
يمكن تعريف الثوابت بطريقة مشابهة تماما لطريقة تعريف المتغيرات ولكن بشرط
1- أن تسبق جملة التعريف بكلمة (const)
2- أن تعطى قيمة ابتدائية (initial value)
3- أن لا نغير قيمتها سواء أكان بجملة الادخال أو بالاسناد حتى لو تم اسناد نفس القيمة الابتدائية لها

الصيغة العامة للتعريف :

```
const DataType constant_name = initial value ;
```

- ملاحظة: (constant_name) تنطبق عليه شروط تسمية المتغيرات.

- أمثلة على تعريف الثوابت :

```
const float tax = 0.80;
const double pi =3.14;
```

رابعا: التعابير الرياضية :
وهي عبارة عن مجموعة من المتغيرات والثوابت والأعداد ترتبط ببعضها البعض عن طريق المعاملات(الإشارات)

- التعابير الرياضية ثلاث أنواع وهي :
1) تعابير حسابية: وهي التي تستخدم المعاملات الحسابية للربط بين أجزاءها ويكون ناتجا عدد .
- المعاملات الحسابية المستخدمة في التعابير الحسابية هي:
(+ الجمع) (– الطرح) (* الضرب) (/ القسمة) (% باقي القسمة الصحيحة)
(++ الزيادة بمقدار واحد) (– – التناقص بمقدار واحد)

- أمثلة على التعابير الحسابية :

```
Sum = n1+ n2;
x = v * by + n – e;
S = (a + b)/5 * t % t2;
```

2) تعابير منطقية: وهي التي تستخدم المعاملات المنطقية للربط بين أجزائها ويكون ناتجا إما صواب (true(1))أو خطأ (false(0))
- المعاملات المنطقية المستخدمة في التعابير المنطقية هي :
(And &&) (Or ||) (Not !)
- أمثلة على التعابير المنطقية:

```
X = n1 && n2;
S = x || !y;
e = s && e || o !x;
```

مثال : اوجد ناتج الجزء التالي من برنامج مكتوب بلغة c++

```
bool x,y,z;
x=true;y=false;
z=x&&y; cout<<z;      الناتج : 0
z=x||y;cout<<z;       الناتج : 1
z=!y;cout<<z;         الناتج : 1
```

3) تعابير مقارنة: وهي التي تستخدم معاملات المقارنة للربط بين أجزائها ويكون ناتجها إما صواب (true (1)) أو (false (0)).

- معاملات المقارنة المستخدمة في التعابير المقارنة هي :
(= = يساوي) (!= لا يساوي) (> اكبر من) (< اصغر من) (>= اكبر من أو يساوي) (<= اصغر من أو يساوي)

- أمثلة على التعابير المقارنة :

```
bool u, X,Y;
int x=5,y=10;
u = x > y;      الناتج : 0
x = x==y;       الناتج : 0
Y = x<=y;       الناتج : 1
```

<u>ملاحظات حول التعابير الرياضية :</u>

1) معامل باقي القسمة (%) يقصد به أن تجد باقي قسمة العدد الموجود على يسار معامل باقي القسمة على العدد الموجود على يمينه ويجب أن يكون ناتج القسمة عدد صحيح ويجب أن تكون القيم الموجودة على يمين معامل باقي القسمة وعلى يساره قيم صحيحة وإلا سينتج لديك خطأ ولن يتم تنفيذ البرنامج.

2) يجب التمييز بين معامل التعيين (=) ومعامل المقارنة (==)
(=): تعني تخزين القيمة الموجودة على يمينه في المتغير على يساره.
(==): تعني هل القيمة الموجودة على يمين هذا المعامل تساوي القيمة الموجودة على يساره.

- (هناك اولويات يجب معرفتها عند تنفيذ التعابير الحسابية وهي التي تحدد العملية التي سوف تنفذ في البداية ثم العملية التي تليها والتي تليها وهكذا) وهي:

1) (الأقواس) (من الداخل للخارج).
2) (الأسس) (القيم التي تكون مرفوعة الأسس).
3) (الضرب) و (القسمة) و (باقي القسمة).
4) (الجمع) و (الطرح).
5) (اكبر من) و (اصغر من) و (اكبر من أو يساوي) و (اصغر من أو يساوي)
6) (يساوي) و (لا يساوي)
7) (معامل التعيين (=))

<u>ملاحظات:</u>
1) عند تساوي الاولويات سيتم التنفيذ من اليسار لليمين
2) في حالة وجود أكثر من معامل تعيين (=) سيتم تنفيذ الجملة من اليمين الى اليسار

مثال: حول العلاقة الرياضية التالية إلى تعبير رياضي بلغة (C++) :

S = (x + y)2v

الحل:

S = (x + y)*2*v;

مثال: بين اولويات تنفيذ العمليات التالية مع إعطاء الناتج النهائي إذا علمت أن (x = 2 , y = 1 , z = 4)

R = (x-y)/1+z*(x%y);

الحل:

1. (2-1)/1+4*(2%1)
2. 1/1+4*(2%1)
3. 1/1+4*0
4. 1+4*0
5. 1+0
6. 1

* <u>هام جدا:-</u>

1) y -=value; , y +=value; , y /=value; , y *=value; جميعها جمل صحيحة وتعني y =y - value; , y =y + value; , y =y / value; , y =y * value; على التوالي.

- تستخدم هذه الجمل عادة للتغير على قيمة المتغير نفسه إما بالزيادة أو بالنقصان.

2) معاملات الزيادة والتناقص (++ و --) تستخدم للزيادة أو تنقيص قيمة المتغير بمقدار واحد

Y++; <u>تعادل</u> y=y + 1; <u>وتعادل</u> y +=1;

y- -; <u>تعادل</u> y=y – 1; <u>وتعادل</u> y -=1;

- إذا كانت الزيادة أو النقصان قبلية (++x/--x) أو بعدية (x++/x--) فلا يؤثر إذا كان المتغير منفرد أما إذا كان المتغير في تعبير حسابي سيؤثر على الناتج.

فإذا كانت الزيادة أو النقصان بعدية (بعد المتغير) في تعبير حسابي تعني التنفيذ ثم الزيادة أو التنقيص للمتغير.

أما إذا كانت الزيادة أو النفصان قبلية (قبل المتغير) في تعبير حسابي تعني الزيادة أو التنقيص لقيمة المتغير قبل التنفيذ.

مثال:

```
int s1=5,s2=10;
cout<<++s1<<endl;        الناتج : 6
cout<<s2++<<endl;        الناتج : 10
s1=s1++ - --s2<<endl;
cout<<s1<<endl;          الناتج : 7
```

الجمل المستخدمة في لغة (++c)

- وهي جمل تقوم كل واحدة منها بوظيفة معينة مخصصة لها.
- سنذكر أهم هذه الجمل لك مع وظيفة كل واحدة والشكل العام(الصيغة العامة) لها مع بعض الملاحظات الهامة.

1) جملة الإخراج (cout)

- الشكل العام (الصيغة العامة)

```
cout<<output;
```

- الوظيفة: إظهار النتائج على الشاشة

- **ملاحظات هامة:**

- المقصود ب output في الصيغة العامة السابقة احد الأمور التالية:
1) ثوابت: مثل : 12 , 3 , 6.0 , 55.89 (أو الثوابت التي نحن نقوم بتعريفها)
2) متغيرات: مثل : a , x , o_e (وفي حالة المتغيرات سيقوم بطباعة قيم هذه المتغيرات).
3) تعابير رياضية: مثل: a+n/(33-0.9) (وفي حالة التعابير سيقوم بطباعة الناتج النهائي للتعبير)
4) نص: مثل: "Hello" (بشرط أن يكون بين إشارتي اقتباس مزدوجة (سيقوم بطباعة ما بين الأقواس)

- يستخدم عادة مع جملة الطباعة كلمة محجوزة وهي endl وهي تستخدم لجعل المؤشر يؤشر على سطر جديد بعد طباعة محتويات جملة الطباعة الموجودة بها.
- يمكن استخدام جملة طباعة واحدة تعمل عمل أكثر من جملة طباعة وذلك باستخدام >> وعليك الانتباه في هذه الحالة إن كان هناك تعابير حسابية سيتم التنفيذ من اليمين الى اليسار ويكون الشكل العام لها:

```
cout<<output_1<<output_2<<.......<<output_n;
```

مثال :

```
int x=5;
cout<<x<<" "<<++x;
```

الناتج: 6 6

- هناك أحرف خاصة تستخدم مع جملة الطباعة للتحكم بشكل المخرجات ويجب أن تكون بين إشارتي اقتباس مزدوجة أو مفرده إذا كانت لوحدها (عند طباعة النصوص) وهذا الجدول يبين لك أهم هذه الحروف:

الحرف	الوصف
\n	يعمل عمل endl ينقل المؤشر إلى بداية سطر جديد
\t	ينقل المؤشر مسافة أفقية للإمام بمقدار خمس فراغات
\\	لطباعة الخط المائل \
\"	لطباعة إشارة الاقتباس المزدوجة "
\'	لطباعة إشارة الاقتباس المفردة '
\r	لنقل المؤشر إلى بداية السطر الحالي
\b	تستخدم لحذف حرف واحد فقط قبل هذا الحرف الخاص

3) جملة الإدخال (cin)

- الشكل العام:

cin>>var;

- الوظيفة: لإعطاء المتغيرات قيم أثناء تنفيذ البرنامج

- ملاحظات هامة:

- المقصود ب var أي متغير تريد إعطاءه قيمة أثناء تنفيذ البرنامج.
- لا يجوز استخدم متغيرات غير معرفة في جملة الإدخال فيجب تعريفها من قبل كما تعلمنا سابقا.
- يمكن استخدام جملة إدخال واحدة تعمل عمل أكثر من جملة وذلك باستخدام >> وفي هذه الحالة سيتم إعطاء المتغيرات قيم ابتداء من اليسار لليمين لكل المتغيرات ويكون الشكل العام لها:

cin>>var_1>>var_2>>………>>var_n;

4) جملة التعيين (=)

- الشكل العام:

Variable_name=expression;

- الوظيفة: تستخدم لإعطاء المتغيرات قيم ولكن أثناء كتابة البرنامج وهذا هو الاختلاف عن جملة الإدخال (cin)

- ملاحظات هامة:

- المقصود ب expression متغير أو ثابت أو تعبير رياضي

- يجب أن يكون المتغير معرف مسبقا ويمكن تعريفه بنفس جملة التعيين ويصبح الشكل العام لها:

Datatype variable_name=expression;

int x=10; (تعريف المتغير وإعطاءه قيمة بنفس الجملة)

int x;
x=10; (تعريف المتغير بجملة وإعطاءه قيمة بجملة أخرى)

5) جملة الشرط (if)

- الشكل العام:

```
If (condition(s))
  Action1(s);
else
  Action2(s);
```

- الوظيفة: تقوم بفحص الشرط فإذا كانت قيمته true(1) يقوم بتنفيذ action1 وإذا كانت قيمته false(0) سيقوم بتنفيذ action2.

- <u>ملاحظات هامة:</u>
- المقصود ب condition(s) هو تعبير حسابي أو منطقي أو مزيج منهما تكون قيمته إما صواب أو خطأ ويجب أن يكون بين قوسين.

- المقصود ب action1 هي الجملة أو الجمل التي سوف تنفذ إذا كان ناتج التعبير true(1) والمقصود ب action2 هي الجملة أو الجمل التي سوف تنفذ إذا كان ناتج التعبير false(0).

- الشكل العام السابق يستخدم عندما نريد أن ننفذ جملة أو جمل في حالة الصواب أو في حالة الخطأ ولكن يمكننا الاستغناء عن الجزء السفلي وعندها سيتم تنفيذ جملة أو جمل في حالة الصواب فقط ويصبح الشكل العام لها كالتالي:

```
If (condition(s))
  Action1;
```

- إذا كان هناك أكثر من جملة واحدة نريد تنفيذها في حالة الصواب أو الخطأ فيجب علينا وضع الجمل داخل اشرتي البداية والنهاية { } .

- إذا كان الشرط عبارة عن متغير واحد فقط فإذا كانت قيمة هذا المتغير غير صفرية يكون الناتج true أما إذا كانت قيمته صفرية(صفر) يكون الناتج false

مثال:

```
int a=5;
If(a)
cout<<"aa"<<endl;
else
cout<<"bb"<<endl;
```

الناتج: aa

- إذا كان الشرط عبارة عن جملة تعيين فيكون قيمته true إذا كانت القيمة المخزنة في المتغير غير صفرية وتكون false إذا كانت القيمة المخزنة به صفرية.

مثال:

```
int a=5;
if(a=7)
cout<<"aa"<<endl;        الناتج: aa
else
cout<<"bb"<<endl;
```

- انتبه لاستخدام (!) فهي تقوم بعكس قيمة التعبير من true إلى false والعكس.

- عند استخدام (&&) بين شرطين أو أكثر يكون ناتج الشرط النهائي true إذا كانت جميع الشروط صحيحة وتكون false إذا كان احدهم على الأقل خطأ.

- عند استخدام (||) بين شرطين أو أكثر يكون ناتج الشرط النهائي true إذا كان احد الشروط صحيح على الأقل ويكون false إذا كانت جميع الشروط خاطئة.

- هناك شكل اخر لجملة (if) وهو :

```
If (condition(s))
       Action1(s);
else if (condition(s))
       Action2(s);
    :
else
      default action;
```

6) جملة الاختيار (switch)

- الشكل العام:

```
switch(expression)
{case exp_1: action1(s);
     break;
 case exp_2: action2(s);
     break;
     :
 case exp_n: action_n(s);
     break;
 default : default_action;}
```

- الوظيفة: تقوم بالتعرف على قيمة expression ثم يقارن قيمته مع القيم الثابتة الموجودة أمام جمل case فإذا كانتا متساويان سيتم تنفيذ الجملة أو الجمل البرمجية الموجودة ضمنها

- ملاحظات هامه:

- المقصود ب expression تعبير له قيمة صحيحة أو متغير والمقصود ب exp قيم ثابتة من نفس نوع expression .

- قد تكون هناك أكثر من جملة ضمن ال case الواحدة وفي هذه الحالة يمكن الاستغناء عن { }

- إن استخدام جزء (Default : default_action;) اختياري فهو يستخدم لتنفيذ جملة أو عدة جمل في حالة عدم تساوي أي من القيم الموجودة أمام كل case وقيمة expression .

- يعد استخدام break اختياري ولكن سوف يتم تنفيذ الجملة أو الجمل الموجودة ضمن case التي قيمته مساوية لقيمة ال expression وجميع الجمل الموجودة أسفل هذه ال case حتى يصل ال default إن كانت موجودة وإذا لم تكن موجودة سيبقى ينفذ حتى الوصول لاخر جملة في switch .او الوصول الى اول جملة break;

- يجوز أن يكون هناك أكثر من exp لل case الواحدة سيتم في هذه الحالة تنفيذ الجملة أو مجموعة الجمل الموجودة ضمن case إذا تساوة قيمة expression مع أي قيمة من قيم exp.

7) جملة الشرط البسيطة(المعامل الشرطي (?:))

- الصيغة العامة:

Condition? expression1: expression2;

- الوظيفة: تقوم بعمل مشابه تماما لعمل جملة (if else) فهي تقوم بتنفيذ Expression1 عندما تكون قيمة ال Condition صحيحة (true) وسيتم تنفيذ expression2 عندما تكون قيمة ال Condition خطأ (false)

- ملاحظات هامة:

- المقصود ب expression1 و expression2 هي الجملة التي سوف تنفذ في حالة صواب اوخطأ الشرط.
- إن عمل (:) مشابه تماما لعمل else في جملة if .
- يستخدم هذا النوع من الجمل عادة في جمل الطباعة وجمل التعيين ويكون شكل الجملة كالتالي لكلا الحالتين

(في جملة الطباعة):

cout<<(condition?expression1:expression2);

مثال :

```
int mark; cin>>mark;
cout<<mark>=50?"pass":"fail";
```

(في جملة التعيين):

Variable_name=condition?expression1:expression2;

مثال :

```
int mark;bool result;
cin>>mark;
result=mark>=50:true:false;
```

8) جملة الدوران (for)

- الصيغة العامة:

```
for (initialization;condition;step)
        Action(s);
```

- الوظيفة: تقوم بفحص قيمة ال condition وبعدها سيتم تنفيذ الجملة أو الجمل التابعة لfor حتى تصبح قيمته خاطئة

- ملاحظات هامة :

- المقصود ب initialization أي تعريف المتغير (يسمى العداد) وإعطاءه قيمة أولية والمقصود ب condition هو شرط استمرار الدوران(تنفيذ الجمل أو الجملة التابعة ل for) والمقصود ب step مقدار الزيادة أو النقصان (التغيير)على العداد بعد كل دورة.

- إذا كان هناك أكثر من جملة نريد تكرارها فيجب وضعها ضمن { } .

- قد تكون الجملة المراد تكرارها في جملة for هي جملة for أخرى وفي هذه الحالة تسمى جملة الدوران المتداخلة ويجب أن يكون العداد في كل جملة for مختلف عن الأخرى .

- إذا تم الاستغناء عن step في جملة for أو استخدامه بشكل خاطئ كأن يجعل قيمة ال condition صواب باستمرار فهذا يؤدي إلى دوران لا نهائي.

9) جملة الدوران (while)

- الصيغة العامة:

```
while (condition(s))
   {action(s);
     step;}
```

- الوظيفة : وظيفتها تكرار تنفيذ جملة أو عدة جمل لعدد معلوم أو غير معلوم من اللفات.

- ملاحظات هامة:

- إذا تم الاستغناء عن step وكانت قيمة ال condition صواب فهذا يؤدي إلى دوران لانهائي.

- يجب تعريف عداد لجملة while خارجها وإعطاءه قيمة أولية ويفضل أن يكون قبل جملة while مباشرة.

- تستخدم جملة الدوران while عادة لدوران غير معلوم عدد مرات دورانه حيث أنها تعتمد على ناتج عملية حسابية أو قيمة مدخلة عن طريق لوحة المفاتيح وفي هذه الحالة(عدد غير معلوم من اللفات) لا داعي لتعريف عداد قبل جملة while ولا داعي لاستخدام step .

10)جملة الدوران (do-while)

- الصيغة العامة:

```
do {
    Action(s);
    Step;     }
while(condition(s));
```

- الوظيفة: تعمل بطريقة مشابهة لجملتي الدوران while و for ولكن تختلف بان الشرط يأتي بأخر الجملة وهذا يعني تنفيذ الجملة أو الجمل التابعة لها عند الدخول إليها بغض النظر عن قيمة ال condition في البداية وسيتم استمرار التنفيذ إذا كانت قيمة ال condition صحيحة true .

- ملاحظات هامة:

- يجب تعريف عداد وإعطاءه قيمة أولية قبل جملة do while فهو الذي سوف تتغير قيمته إما بالزيادة أو النقصان بواسطة استخدام step .

- سيتم تنفيذ الجملة أو الجمل التابعة لجملة do while مرة واحدة على الأقل.

- يجب أن تنتهي جملة do while بفاصلة منقوطة.

- إذا كان هناك أكثر من جملة مراد تكرارها يجب وضعها ضمن { } .

////// ملاحظات عامة وهامة جدا جدا ///////

1) عند كتابتك أي برنامج باستخدام لغة ++c يجب أن يبدأ برنامجك

```
#include<iostream.h>
void main(){
    برنامجك يكتب هنا
}
```

2) أي خطأ بسيط قد ترتكبه أثناء كتابتك للبرنامج سيؤدي إلى عدم تنفيذ البرنامج أو سيتم تنفيذه وتحصل على نتائج غير متوقعة.

3) من الأخطاء التي قد ترتكبها أثناء كتابتك لبرنامج باستخدام لغة (++c):

- أن تنهي جملة **if** بفاصلة منقوطة كالتالي **;(if (condition(s)** من دون أن تحتوي على **else** وفي هذه الحالة سوف يتم تنفيذ البرنامج وسوءا إن كان الشرط صحيح أو خطأ فسوف يتم تنفيذ الجملة أو الجمل التابعة لها.

- أن تنهي جملة **if** بفاصلة منقوطة كالتالي **;(if (condition(s)** وتحتوي على **else** في هذه الحالة لا يتم تنفيذ البرنامج لان **else** في هذه الحالة ليس لها **if**.

- أن تنهي جملة **for** أو جملة **while** بفاصلة منقوطة وفي هذه الحالة سوف يتم تنفيذ البرنامج ولكن سوف ينتج لدينا خطأ في النتائج.
- أن نفصل بين مكونات جملة **for** بفواصل بدلا من الفواصل المنقوطة
- استخدام متغير أو متغيرات من دون تعريفها مسبقا.

- أن تعرف متغير وتقوم بتسميته باستخدام الأحرف الصغيرة وتقوم باستخدامه لاحقا بنفس الاسم ولكن باستخدام الحروف الكبيرة او بالعكس.

- أن تقوم بتخزين قيمة غير صحيحة في متغير لتخزين القيم الصحيحة (int) وفي هذه الحالة سوف يقوم بتخزين الجزء الصحيح فقط في المتغير.

- يجب أن تكتب الكلمات المحجوزة بحروف صغيره.

- عدم إنهاء الجمل بفاصلة منقوطة في الجمل التي يجب إنهائها بفاصلة منقوطة(جمل التعريف والتعيين.........).

- أن تستخدم قوس واحد فقط ولا تقوم بتسكيره بقوس اخر.

4) يجب الانتباه لاستخدام جملة **;break** و جملة **;continue** في جمل الدوران
- **;break** ← (تقوم بالخروج من الدوران مباشرة)
- **;continue** ← (سيتم التجاهل عن الجملة أو الجمل التي تليها والانتقال مباشرة للدورة التي تليها)

- تستخدم عادة جملة **break** و جملة **continue** في جمل الدوران باستخدام جمل الشرط.

إليك هذه المجموعة من الأمثلة مع حلولها:-

1) اكتب برنامج يقوم بإدخال عدد صحيح ثم يقوم بطباعة مربع العدد ومكعبه على الشاشة.

2) اكتب برنامج يقوم بحساب مساحة المثلث ومساحة المستطيل علما أن:
مساحة المثلث = 0.5 * القاعدة * الارتفاع
مساحة المستطيل = الطول * العرض

3) اكتب برنامج يقوم بإدخال عدد صحيح فإذا كن العدد موجب وزوجي يقوم بطباعة العبارة التالية Positive and even وإذا كان غير ذلك يقوم بطباعة Error.

4) اكتب برنامج يقوم بإيجاد العلاقة التالية:

f(n,x)= nx-3 n>x
n+x n=x
x+2x n<x

5) اكتب برنامج يقوم بإدخال حرف فإذا كان الحرف a يقوم بطباعة كلمة hi وإذا كان الحرف b يقوم بطباعة كلمة yes وإذا غير ذلك يقوم بطباعة كلمة error .

6) اكتب برنامج يقوم بإدخال عددين صحيحين وثم طباعة العدد الأكبر.

7) اكتب برنامج يقوم بإدخال عدد صحيح ويقوم البرنامج بطباعة مضروب هذا العدد.

8) اكتب برنامج لإيجاد مجموع الإعداد الزوجية من 1 إلى 100.

9) اكتب برنامج يقوم بإدخال أعداد صحيحة إذا كانت لا تساوي 5 يقوم بجمعها وإذا تم إدخال العدد 5 يقوم بطباعة ناتج جمع الأعداد التي تم إدخالها.

10) اكتب برنامج يقوم بإدخال 30 عدد صحيح وطباعتها إلا الأعداد الزوجية فلا يقوم بطباعتها

11) اكتب برنامج لطباعة الشكل التالي باستخدام جمل الدوران

```
#
##
###
####
```

12) اكتب برنامج لإيجاد ناتج المتسلسلة التالية:

X=1/1!+2/2!+3/3!

13) اكتب برنامج لإيجاد ناتج المتسلسلة التالية:

sum=3/(3+5) + 5/(5+7) + 7/(7+9) + ……………… +19/(19+21)

14) اكتب برنامج يقوم لإيجاد ناتج المتسلسلة التالية:

x=3/10 + 4/17 + 5/26 + 7/50 + ……. + 10/101

15) اكتب برنامج لإيجاد ناتج المتسلسلة التالية:

sum= (2*2)/(3*3) + (4*4)\(5*5) + (6*6)\(7*7) + 1/2 + 2/5 + 3/10 +4/17

(1)	(2)
#include<iostream.h> void main() { int n; cout<<"Enter Number:\n"; cin>>n; cout<<"square is:"<<n*n<<endl; cout<<"cubic is:"<<n*n*n<<endl; }	#include<iostream.h> void main() { double base,high; int length,width; cin>>base>>high; cout<<"triangle area is:\n" cout<<0.5 * base * high<<endl; cin>>length>>width; cout<<"rectangle area is:\n" cout<<length*width<<endl; }
(3)	**(4)**
#include<iostream.h> void main() { int n; cout<<"Enter Number:"<<endl; cin>>n; if ((n >= 0) && (n%2==0)) cout<<" Positive and even"<<endl; else cout<<"Error\n"; }	#include<iostream.h> void main() { int n,x,y; cin>>n>>x; if(n>x) {f=n*x-3; cout<< f<<endl;} else if(n==x){ f=n+x; cout<<f<<endl;} else{ f=x+2*x; cout<<f<<endl;}}
(5)	**(6)**
#include<iostream.h> void main() { char c; cin>>c; switch(c) { case 'a': cout<<"hi"<<endl; break; case 'b': cout<<"yes"<<endl; break; default: cout<<"error"<<endl; }}	#include<iostream.h> void main() { int n1,n2,max; cin>>n1>>n2; max = n1>n2 ? n1 : n2; cout<<max<<endl; }

(7)

```
#include<iostream.h>
void main()
{
int x,p=1;
cout<<"Enter Number:"<<endl;
cin>>x;
for ( int i =0;i <= x ; i++)
p*=i;
cout<<p<<endl;
}
```

(8)

```
#include<iostream.h>
void main()
{
int i=2;int sum=0;
while ( i<=100 ){
sum+=i;
i+=2;
}
cout<<sum<<endl;
}
```

(9)

```
#include<iostream.h>
void main()
{
int n,sum=0;
cout<<"Enter Number:"<<endl;
cin>>n;
while ( n !=5 ){
sum+=n;
cout<<"Enter Number:"<<endl;
cin>>n;
}
cout<<sum<<endl;
}
```

(10)

```
#include<iostream.h>
void main()
{
int n;
for (int i=1; i<=30 ; i++){
cout<<"Enter Number:\n";
cin>>n;
if ( n%2==0 )
continue;
else
cout<<n<<endl;}
}
```

(11)

```
#include<iostream.h>
void main()
{
for ( int i=1;i<=4;i++ )
{
for ( int j=1;j<=i;j++ )
cout<<"#";
cout<<endl;}
}
```

(12)

```
#include<iostream.h>
void main()
{
int i;
float x=0.0;
for (i=1;i<=3;i++)
{
 int p=1;
for( int j=1;j<=i;j++ )
p*=i;
x+=i/p;
}
cout<<x<<endl;
}
```

(13)	(14)
#include<iostream.h> void main() { float sum=0.0; for (int i=3;i<=19;i+=2) sum+=i/(i+(i+2)); cout<<sum<<endl; }	#include<iostream.h> void main() { int x=0; for(int i=3;i<=10;i++) if (i==6) continue; else sum+=i/(i*i+1); cout<<x<<"\n"; }
(15)	
#include<iostream.h> void main() { float sum=0.0; for(int i=2;i<=7;i+=2) sum+=(i*i)\((i+1)*(i+1)); for(int j=1;j<=4;j++) sum+=j\(j*j+1); cout<<sum<<endl; }	

المصفوفات (Arrays):

ـ المصفوفة: نوع من أنواع تراكيب البيانات وتتكون من مجموعة من البيانات من نفس النوع وتكون على شكل مواقع متجاورة في الذاكرة لها نفس الاسم ولكن لكل موقع مرجع أو رقم مخصص له للتعامل مع القيم المخزنة في هذه المواقع.

ـ قد تتساءل لماذا نستخدم المصفوفات ولماذا لا نخزن القيم بمتغيرات مثلا نحن نقوم بتعريفها؟؟
قد يكون تساؤلك في محله ولاكن أنا أزيدك علما بان المصفوفات تسهل عليك الكثير من العمليات والأمور التي تحتاجها ويحتاجها كل مبرمج في برنامجه.

المصفوفات نوعان:
1) مصفوفات أحادية البعد 2) مصفوفات ثنائية البعد

((ملاحظات على المصفوفات))

- عند تعريف مصفوفة فان كل موقع فيها سيأخذ رقم مخصص به ويبدأ الترقيم من القيمة 0 إلى (حجم المصفوفة(نحن نقوم بتحديده)-1)وذلك بالتسلسل ....0,1,2 هذا في المصفوفات الأحادية أما في المصفوفات الثنائية فسوف يبدأ الترقيم من 0 إلى عدد الأعمدة-1 في البعد الأول وفي البعد الثاني سوف يبدأ الترقيم من 0 إلى عدد الصفوف-1.

- نتعامل مع القيم في المصفوفة باستخدام اسم المصفوفة ورقم الموقع الذي توجد به القيمة بين [] [رقم الموقع].

1) المصفوفات أحادية البعد (One-Dimension)

- نقصد بأحادية البعد أي أنها تتكون من صف واحد(عمود واحد) من المواقع المتجاورة.

- لكي نستخدم مصفوفة أحادية البعد يجب تعريفها كالتالي:
datatype arrayname[size];

- datatype: هو نوع البيانات للقيم التي سوف يتم تخزينها في المصفوفة مثل(int,float,char,double,..)
- arrayname: وهو اسم المصفوفة ويجب أن نراعي شروط تسمية المتغيرات في تسمية المصفوفات.
- [size]: حجم المصفوفة الذي تريده(عدد المواقع التي نريد التخزين بها).
- ولتوضيح الأمر أكثر سأقوم برسم توضيحي لمعرفة الكيفية التي نتعامل بها مع المصفوفات أحادية البعد:
لو فرضنا أن لدينا المصفوفة التالية: int a[5];

	القيمة	رقم الموقع
a[0]	22	0
a[1]	-5	1
a[2]	0	2
a[3]	10	3
a[4]	3	4

- نلاحظ أن الموقع الأخير في المصفوفة يحمل رقم 4 وهو(size-1).
- نلاحظ أن الموقع الأول في المصفوفة يأخذ رقم (0).
- إذا أردنا القيمة (10) مثلا فإننا نستخدم a[3] وإذا أردنا القيمة (0) نستخدم a[2] وهكذا....

- يجب التعبير عن رقم الموقع برقم صحيح أو بتعبير رياضي ينتج عنه رقم صحيح فمثلا: a[2+1] تكافىء a[3] ويقصد به القيمة الموجودة في هذا الموقع وهي 10.
ولو فرضنا int x=2; وقمنا باستخدام الجملة التالية cout<<a[x+1]; فانه سيتم طباعة القيمة الموجودة الموقع (3) a[3]; التي هي (10) ولكن عليك الانتباه من هذه الجملة: a[0]+=4; فهذه لا تعني القيمة الموجودة في الموقع(4) بل تعني زيادة 4 على القيمة الموجودة في الموقع(0) فتصبح حسب المثال السابق 26

* ملاحظة:
- نستخدم عادة جمل الدوران لكي يسهل علينا التعامل مع المصفوفات حيث يمكن استخدام عداد الدوران كدليل لأرقام المواقع الموجودة في المصفوفة.
- تستخدم عادة جملة دوران واحدة للقيام بعمل معين في حالة التعامل مع المصفوفات أحادية البعد.

- بناءا على الرسم التوضيحي السابق سنقوم بالتعرف على كيفية استخدام جمل الدوران مع مصفوفة أحادية البعد:

```
for(int i=0;i<5;i++)
 cout<<a[i]<<" ";
```

- في جملة الدوران السابقة قمنا باستغلال عداد الدوران(i) للتعامل مع مواقع المصفوفة المختلفة وتغيير قيمة العداد تعمل على التنقل بين تلك المواقع.

- في البداية ستكون قيمة العداد 0 وتكون جملة الطباعة cout<<a[0]<<" "; وسيقوم بطباعة القيمة الموجودة في ذلك الموقع وهي 22.

بعدها سيتم زيادة قيمة العداد بمقدار واحد وتصبح قيمته 1 وجملة الطباعة تصبح cout<<a[1]<<" "; وسيقوم بطباعة القيم الموجودة في تلك الموقع وهي (-5) وهكذا إلا أن تصل قيمة العداد 5 سيتم الخروج من الدوران وسيكون الشكل النهائي بعد تنفيذ جملة الطباعة السابقة كالتالي: 22 -5 0 10 3

- عندما نقوم بتعريف المصفوفة سوف يتم حجز مواقع لها في الذاكرة حسب حجمها فإذا كان حجمها مثلا 10 سيتم حجز 10 مواقع لها في الذاكرة ونستطيع إدخال القيم في هذه المواقع بواسطة جملة الإدخال كالتالي:

```
for(int i=0;i<10;i++)
 cin>>a[i];
```

سيتم التعامل مع جملة الإدخال تماما مثل جملة الطباعة فعندما تكون قيمة العداد في البداية 0 ستكون جملة الإدخال كالتالي: cin>>a[0]; وسيتم إدخال قيمة وهكذا لجميع المواقع في المصفوفة وبعد إتمام تنفيذ جملة الدوران كاملة تكون جميع القيم التي قمنا بإدخالها مخزنة في مواقع المصفوفة وبالترتيب من الموقع الأول إلى الموقع الأخير.

--- نستطيع أن نعرف مصفوفة أحادية البعد وإعطاءها قيم أولية وذلك كالتالي:

```
datatype arrayname[size]={value_1,value_2,………,value_size};
```

- **ملاحظات:**

- **تحديد حجم المصفوفة في هذه الحالة يكون اختياري(يجوز تحديده ويجوز عدم تحديده) ويكون عدد العناصر المحصورة بين{} هي حجم المصفوفة.**

- **عند تحديد حجم المصفوفة في الحالة السابقة يجب أن يكون حجمها اكبر أو يساوي من عدد القيم المخزنة فيها(المحصورة بين {}).**

- **اذا كان حجم المصفوفة اقل من القيم المخزنة داخل {} يعطي خطأ اما اذا كان حجمها اكبر فيتم اعطاء باقي القيم للمصفوفة أصفار.**

- **يكون رقم موقع value_1 (0) ورقم موقع value_2 (1) وهكذا........**

- **انظر إلى هذا المثال على تعريف مصفوفة وإعطاءها قيم أولية مع رسم توضيحي له لكي تتضح لك الفكرة أكثر:**

int a[]={1,5,0,-8}; **(لم يتم تحديد حجم المصفوفة ولكن نستطيع معرفته من خلال عدد عناصرها وهو 4)**
int a[4]={1,5,0,-8}; **(تم تحديد حجم المصفوفة بناءا على عدد عناصرها)**

الموقع	0	1	2	3
القيمة	1	5	0	-8

- **هذه مجموعة من الأمثلة على المصفوفات أحادية البعد مع حلولها وشرحها:-**

1) اكتب برنامج يقوم بقراءة 5 قيم صحيحة وحفظها في مصفوفة ثم طباعة محتويات تلك المصفوفة ثم يقوم بجمع قيم هذه المصفوفة وطباعة ناتج الجمع.

#include<iostream.h> void main(){ int x[5];int sum=0;int i; for(i=0;i<5;i++) cin>>x[i]; for(i=0;i<5;i++) cout<<x[i]<<endl; for(i=0;i<5;i++) sum+=x[i]; cout<<sum<<endl; }	عرفنا مصفوفة أحادية البعد اسمها x وحجمها 5 ثم قمنا باستخدام ثلاثة جمل دوران الأولى لكي نقوم بإدخال القيم إلى المصفوفة والثانية لطباعة عناصر المصفوفة بشكل عمودي وذلك بسبب استخدام endl واستخدمنا جملة الدوران الثالثة لكي تقوم بجمع جميع القيم في هذه المصفوفة وتخزينها في المتغير sum الذي قمنا بتعريفه في البداية وأعطيناه قيمة أولية 0 لكي لا تؤثر على ناتج الجمع وبعد ذلك يقوم البرنامج بطباعة ناتج الجمع لجميع القيم المخزنة في المصفوفة.

2) اكتب برنامج يقوم بإدخال 10 قيم صحيحة إلى مصفوفة وثم إيجاد اكبر قيمة واصغر قيمة في هذه المصفوفة.

#include<iostream.h> void main(){ int ar[10],i,j,max,min; for(i=0;i<10;i++) cin>>ar[i]; max=min=ar[0]; for(j=1;j<10;j++) {if(ar[j]>max) max=ar[j]; if(ar[j]<min) min=ar[j];} cout<<max<<endl; cout<<min<<endl; }	عرفنا مصفوفة أحادية البعد اسمها ar حجمها 10 وقمنا أيضا بتعريف متغيرين الأول اسمه max وذلك لنخزن به اكبر قيمة في المصفوفة ومتغير آخر اسمه min لنخزن به اصغر قيمة في المصفوفة وبعدها تم استخدام جملة دوران لنقوم بإدخال القيم وتخزينها في المصفوفة وبعد الانتهاء من إدخال القيم نقوم بتخزين القيمة الموجودة في الموقع الأول(القيمة الأولى) في المتغير max والمتغير min على فرض أنها اكبر قيمة واصغر قيمة في المصفوفة وذلك لمقارنتها مع باقي قيم المصفوفة واستخدمنا جملة دوران أخرى ولكن عدادها يبدأ من 1 بدلا من 0 لأنه سيقارن 9 قيم ابتدءا من القيمة الثانية مع القيمة الأولى التي قمنا بتخزينها في المتغيرين max و min فإذا كان ناتج جملة if الأولى صواب true تخزن القيمة الجديدة(الأكبر) في المتغير max وإذا كان ناتج جملة if الثانية true تخزن القيمة الجديدة(الأصغر) في min ثم يقوم بطباعة اكبر قيمة وتليها اصغر قيمة.

3) اكتب برنامج يقوم بتخزين جدول الضرب للعدد 6 في مصفوفة ثم يقوم بطباعته.

#include<iostream.h> void main(){ int array[11];int i; for(i=0;i<=10;i++) array[i]=6*i; for(i=0;i<=10;i++) cout<<array[i]<<endl; }	عرفنا مصفوفة أحادية البعد حجمها 11 لان جدول الضرب يتكون من 11 قيمة ابتدءا من (6*0) حتى (6*10) واستخدمنا جملة الدوران الأولى لكي نقوم بتخزين قيم جدول الضرب للعدد 6 في المصفوفة التي تنتج من ضرب العدد 6 بالعداد i الذي يبتدئ بقيمه أولية 0 حتى ينتهي وقيمته النهائية 10 واستخدمنا بعد ذلك جملة دوران أخرى لكي تقوم بطباعة الناتج وذلك بشكل عمودي.

4) اكتب برنـامج يقوم بتعريف مصفوفة تحتوي على القيم التالية (5,8,75,19,30) وبعدها يقوم بطباعة هذه القيم وبجانب كل قيمة عدد من النجوم مساوي للقيمة .

#include<iostream.h> void main(){ int s[5]={5,8,46,19,30};int i,j; for(i=0;i<5;i++) {cout<<s[i]<<" "; for(j=0;j<s[i];j++) cout<<"*"; cout<<endl;} }	**عرفنا مصفوفة أحادية البعد وقمنا بتخزين 5 قيم أولية فيها وهي القيم المعطاة في السؤال فيكون إذا حجم المصفوفة 5 بناءا على القيم المخزنة بها ثم استخدمنا جملتين دوران الأولى رئيسية تقوم بطباعة القيم المخزنة في المصفوفة وتحتوي على جملة دوران أخرى(جملة الدوران الثانية الداخلية) تقوم بطباعة نجوم بجانب كل قيمة بعدد مساوي للقيمة وذلك باستخدام j<s[i] .ونلاحظ استخدام cout<<endl; فهذه الجملة تتبع لجملة الدوران الرئيسية حيث أنها تقوم بعد طباعة القيمة وبجانبها عدد النجوم بنقل المؤشر إلى السطر التالي.**

5) اكتب برنـامج يقوم بقراءة 10 قيم صحيحة من لوحة المفاتيح ويقوم بتخزين الإعداد الزوجية في مصفوفة اسمها even والإعداد الفردية في مصفوفة اسمها odd.

#include<iostream.h> void main(){ int even[10];int odd[10];int i,n,c1=0,c2=0; for(i=0;i<10;i++) {cin>>n; if(n%2==0) { even[c1]=n; c1++;} else { odd[c2]=n; c2++;}} for(i=0;i<c1;i++) cout<<even[i]<<endl; cout<<endl; for(i=0;i<c2;i++) cout<<odd[i]<<endl; }	**قمنا بتعريف مصفوفتين الأولى اسمها even لتخزين القيم الزوجية وهي أحادية البعد والثانية لتخزين القيم الفردية واسمها odd وكلا المصفوفتين لها الحجم 10 لان اكبر عدد ممكن من القيم أن يخزن في أحداهما 10 قيم وعرفنا متغيرين c1 و c2 وكلاهما له قيمة أولية 0 فالمتغير c1 لتخزين القيم في المصفوفة الزوجية في الموقع الأول في البداية لان قيمته الأولية 0 والمتغير c2 لتخزين القيم في المصفوفة الفردية في الموقع الأول في البداية لان قيمته الأولية 0 وبعدها استخدمنا جملة دوران تقوم بإدخال قيمة وفحص هذه القيمة زوجية أم فردي باستخدام جملة if فإذا كانت القيمة زوجية تخزن في المصفوفة even في الموقع الأول باستخدام المتغير c1 ومن ثم زيادة قيمة هذا المتغير بمقدار واحد لكي يقوم بتخزين القيمة الجديدة في الموقع الثاني 1 أما إذا كانت القيمة غير زوجية(فردية) تخزن في المصفوفة odd في الموقع الأول باستخدام المتغير c2 ومن ثم زيادة قيمة هذا المتغير بمقدار واحد لكي يخزن القيمة الجديدة في الموقع الثاني 1 وهكذا.ثم استخدمنا جملة دوران أخرى لكي تقوم بطباعة عناصر المصفوفة الزوجية بشكل عمودي واستخدمنا جملة cout<<endl; لفصل المصفوفة الزوجية عن الفردية عند الطباعة وبعدها استخدمنا جملة دوران ثالثة لطباعة عناصر المصفوفة الفردية بشكل أيضا عمودي.**

6) اكتب برنامج يقوم بإدخال 8 قيم إلى مصفوفة وعند البحث عن أي قيمة في المصفوفة يطبع Found إذا كانت موجودة ويطبع العدد وبجانبه موقعه في المصفوفة وإذا لم تكن القيمة مخزنة يطبع Not Found .

قمنا بتعريف مصفوفة أحادية البعد حجمها 8 وعرفنا متغير اسمه found من نوع bool (أي قيمة هذا المتغير إما true أو false) وخزنا به قيمة أولية false ثم استخدمنا جملة دوران لإدخال القيم في المصفوفة واستخدمنا جملة دوران أخرى لكي نقوم بفحص وجود القيمة التي قمنا بإدخالها وذلك باستخدام جملة if التابعة لها فإذا كان جواب الشرط true سوف يغير قيمة المتغير found وتصبح قيمته true وطباعة كلمة Found وطباعة القيمة وبجانبها موقعها في المصفوفة الذي يحدده العداد i ثم نستخدم جملة if أخرى لا تتبع لجملة الدوران لكي نقوم بفحص قيمة المتغير found فإذا بقيت على قيمتها الأولية false فهذا يعني أن القيمة غير موجودة في المصفوفة فيقوم بطباعة Not Found وإذا تغيرت قيمة المتغير وأصبحت true فهذا يعني أن القيمة التي تم إدخالها مخزنة في المصفوفة أي أنها وجدت.	#include<iostream.h> void main(){ int ax[8],i,n;bool found=false; for(i=0;i<8;i++) cin>>ax[i]; cout<<"Enter Value:"<<endl; cin>>n; for(i=0;i<8;i++) if(n==ax[i]){ found=true; cout<<"Found"<<endl; cout<<ax[i]<<" "<<i<<endl;} if(found==false) cout<<"Not Found"<<endl; }

7) اكتب برنامج يقوم بقراءة 5 قيم صحيحة وتخزينها في مصفوفة ثم يقوم بعكس عناصر المصفوفة(جعل القيمة الموجودة في الموقع الأول في الموقع الأخير والقيمة الموجودة في الموقع الثاني في الموقع قبل الأخير وهكذا..) وبعدها يقوم بطباعة المصفوفة.

يقوم هذا البرنامج بعكس تخزين عناصر المصفوفة ويختلف عن طباعة المصفوفة بالعكس فإذا أردنا أن نطبع المصفوفة بالعكس نستخدم جملة دوران للطباعة تكون قيمة العداد الأولية 4 لأنه أخر موقع فيها ويكون شرط استمرار الدوران i>=0 لأن 0 أول موقع ويتغير العداد بالنقصان.أما في مثالنا هذا بعد استخدام جملة الدوران لإدخال القيم إلى المصفوفة تم تعريف متغير c وخزنا فيه قيمة أولية 4 لأنه يعتبر أخر موقع في المصفوفة واستخدمنا جملة دوران لكي تقوم بتخزين(في البداية) القيمة الموجودة في الموقع الأول في المصفوفة في متغير t وتخزين القيمة الموجودة في الموقع الأخير في أول موقع في المصفوفة ثم نقلنا القيمة الموجودة في المتغير t(القيمة الموجودة في أول موقع في بداية تنفيذ البرنامج) إلى أخر موقع في المصفوفة ثم قمنا بتنقيص قيمة المتغير c لكي نقوم بنفس العمل مع باقي عناصر المصفوفة ثم قمنا بطباعة عناصر المصفوفة بشكل عمودي باستخدام جملة دوران.	#include<iostream.h> void main(){ int arr[6],i,t,c=5; for(i=0;i<6;i++) cin>>arr[i]; for(i=0;i<3;i++) {t=arr[i]; arr[i]=arr[c]; arr[c]=t; c--;} for(i=0;i<6;i++) cout<<arr[i]<<endl; }

8) اكتب برنامج يقوم بإدخال 10 قيم صحيحة من لوحة المفاتيح ويخزنها في مصفوفة ثم يقوم بترتيبها تنازليا(من القيمة العظمى إلى القيمة الصغرى).

عرفنا مصفوفة أحادية البعد حجمها 10 وعرفنا متغير u لكي يساعدنا في عملية الترتيب واستخدمنا جملة دوران لإدخال القيم في المصفوفة وجملة دوران أخرى ولكن شرط الدوران فيها j<9 ولم نكتبه j<10 لأننا عندما نصل إلى الموقع 8 سوف نقارن القيمة الموجودة فيه مع القيمة الموجودة في الموقع 9 (الموقع الأخير) وذلك حسب جملة الدوران الثانية التابعة لجملة الدوران الأولى(الرئيسية) وتكون عملية المقارنة بواسطة جملة if التي تقوم بفحص القيمة في الموقع الحالي مع القيمة المخزنة في الموقع الذي يليه فإذا كانت القيمة في الموقع الحالي اصغر من القيمة في الموقع الذي يليه تخزن القيمة الأكبر في المتغير u وتخزن القيمة الأصغر في الموقع الذي يلي الموقع الحالي وبعدها نقوم بتخزين القيمة الموجودة في المتغير u (القيمة الأكبر) في الموقع الحالي الذي يكون في البداية الموقع الأول 0 وهكذا مع باقي قيم المصفوفة ثم استخدمنا جملة دوران لنقوم بطباعة عناصر المصفوفة بعد ترتيبها بشكل عمودي.

```
#include<iostream.h>
void main(){
int a[10],i,j,u;
for(i=0;i<10;i++)
cin>>a[i];
for(j=0;j<9;j++){
for(i=j+1;i<10;i++)
if(a[j]<a[i]){
u=a[i];
a[i]=a[j];
a[j]=u;}}
for(i=0;i<10;i++)
cout<<a[i]<<endl;
}
```

9) اكتب برنامج يقوم بتعريف مصفوفتين وإدخال 5 قيم في كل مصفوفة وفحص إذا كانتا المصفوفتين متساويتين(كل عنصر في المصفوفة الأولى يساوي العنصر في المصفوفة الثانية في القيمة والموقع) يطبع Equal وإذا كانتا غير متساويتان يطبع Not Equal .

عرفنا مصفوفتين الأولى اسمها a حجمها 5 وهي أحادية البعد والثانية اسمها b وحجمها 5 وهي أيضا أحادية البعد وقمنا بتعريف متغير اسمه e من نوع bool واستخدمنا جملتين دوران الأولى لإدخال القيم في المصفوفة a والأخرى لإدخال القيم في المصفوفة b واستخدمنا بعدها جملة دوران أخرى تستخدم جملة if لمقارنة أول قيمة في المصفوفة a مع القيمة الأولى في المصفوفة b فإذا كانت القيمتين متساويتين نخزن true في المتغير e وننتقل إلى الدورة التي تليها لمقارنة القيمتين التاليتين وهكذا إلى أن نصل إلى أخر قيمة في المصفوفة a فنقارنها مع القيمة الأخيرة في المصفوفة b وبعدها يخرج من الدوران وبعد الخروج من الدوران نستخدم جملة if لمعرفة إذا كانت قيمة المتغير e true يقوم بطباعة Equal أما إذا كانت القيمة الأولى في المصفوفة a لا تساوي القيمة الأولى في المصفوفة b أو لم تتساوى أي قيمة من قيم المصفوفة a مع القيمة المخزنة في المصفوفة b تغيير قيمة المتغير e وتصبح false ونستخدم جملة break لكي نخرج مباشرة من الدوران والانتقال إلى جملة if التي تلي جملة الدوران وفحص قيمة المتغير e إذا كانت trueيقوم بطباعة Equal أما إذا كانت false يقوم بطباعة Not Equal.

```
#include<iostream.h>
void main(){
int a[5],b[5],i;bool e;
for(i=0;i<5;i++)
cin>>a[i];
for(i=0;i<5;i++)
cin>>b[i];
for(i=0;i<5;i++)
if(a[i]==b[i])
e=true;
else
{e=false;
break;}
if(e==true)
cout<<"Equal"<<endl;
else
cout<<"not Equal"<<endl;
}
```

10) اكتب برنامج يقوم بإدخال 5 قيم صحيحة إلى مصفوفة ونسخ هذه القيم إلى مصفوفة أخرى وثم إدخال قيمة إذا كانت مخزنة في المصفوفة الجديدة يقوم بطباعة القيمة والقيمة التي تليها والتي قبلها وإذا لم تكن مخزنة يطبع Error.

عرفنا مصفوفتين الأولى اسمها a وحجمها 5 وهي أحادية البعد والثانية اسمها b وحجمها 5 وهي أيضا أحادية البعد وعرفنا متغير من نوع bool اسمه found واستخدمنا جملة دوران لتقوم بإدخال القيم إلى المصفوفة a ثم استخدمنا جملة دوران أخرى لنقل القيم الموجودة في المصفوفة a إلى المصفوفة b وذلك بالترتيب من الموقع الأول إلى الموقع الأخير في المصفوفة ثم قمنا بإدخال قيمة لكي يقوم البرنامج بمقارنتها مع قيم المصفوفة b فإذا كانت القيمة موجودة في المصفوفة يقوم بجعل قيمة المتغير found <u>true</u> فيقوم بطباعة القيمة والقيمة التي تليها والقيمة التي تسبقها ثم يخرج من الدوران مباشرة بسبب استخدام break(مع ملاحظة إذا قمنا بإدخال القيمة تساوي للقيمة الموجودة في الموقع الأخير يقوم بطباعة القيمة التي قبله وطباعة القيمة وطباعة القيمة التي بعده(التي تكون القيمة الموجودة في أول موقع في المصفوفة) وإذا قمنا بإدخال قيمة مساوية للقيمة الموجودة في الموقع الأول في المصفوفة فانه سيقوم بطباعة 0 قبل القيمة ومن ثم طباعة القيمة وطباعة القيمة التي تليها) أما في حالة كانت القيمة المدخلة لا تساوي أي قيمة من قيم المصفوفة فانه سيجعل قيمة المتغير found <u>false</u> وعند الخروج من الدوران يقوم بطباعة Error.	`#include<iostream.h>` `void main(){` `int a[5],b[5],i,v;bool found=false;` `for(i=0;i<5;i++)` `cin>>a[i];` `for(i=0;i<5;i++)` `b[i]=a[i];` `cout<<"Enter Value:"<<endl;` `cin>>v;` `for(i=0;i<5;i++)` `if(v==b[i])` `{found=true;` `cout<<b[i-1]<<" "<<b[i]<<" "<<b[i+1]<<endl;` `break;}` `if(found==false)` `cout<<"Error"<<endl;` `}`

2) المصفوفات ثنائية البعد (Two-Dimension)

- تتكون من مجموعة من الصفوف ومجموعة من الأعمدة.

- لكي نستخدم مصفوفة ثنائية البعد يجب تعريفها كالتالي:

datatype arrayname[rows][columns];

- datatype: هو نوع البيانات للقيم التي نريد تخزينها في المصفوفة.
- arrayname: اسم المصفوفة ويجب أن نراعي شروط تسمية المتغيرات في تسمية المصفوفات.
- [rows]: عدد الصفوف في المصفوفة ثنائية البعد.
- [columns]: عدد الأعمدة في المصفوفة ثنائية البعد.
- ولتوضيح الأمر أكثر سأقوم برسم توضيحي لمعرفة كيفية التعامل مع المصفوفات(ثنائية البعد) :

لو فرضنا أننا عرفنا المصفوفة التالية التي اسمها **a** ونوع البيانات المخزنة بها أعداد صحيحة **int a[2][3];**

* فإذا أردنا القيمة (3) مثلا فإننا نستخدم ;[2][0]a فالقيمة 3 موجودة في تقاطع الصف الأول الذي رقمه 0 والعمود الثالث الذي رقمه (2) وإذا أردنا القيمة (7) فإننا نستخدم ;[0][0]a لان القيمة 7 موجودة في الصف الأول والعمود الأول وهكذا....

رقم الصف والعمود	0	1	2
0	7	90	3
1	12	5	22

- ملاحظة(1):- تخضع أرقام الصفوف والأعمدة في المصفوفات ثنائية البعد لنفس الشروط التي تم تحديدها في المصفوفات أحادية البعد.
- ملاحظة(2): - تستخدم عادة جملة دوران متداخلة للقيام بعمل معين عندما تكون المصفوفات ثنائية البعد حيث تكون جملة الدوران الرئيسية للصفوف أما جملة الدوران الداخلية فتكون للأعمدة.

- بناءا على الرسم التوضيحي السابق سنتعرف على كيفية التعامل مع المصفوفات ثنائية البعد:

إذا أردنا أن نقوم بطباعة عناصر المصفوفة السابقة نستخدم جملتين دوران الأولى لكي تتنقل بين الصفوف والأخرى لكي تتنقل بين الأعمدة في كل صف كالتالي:

```
for(int i=0;i<2;i++)
for(int j=0;j<3;j++)
cout<<a[i][j];
```

- في البداية تكون قيمة العداد i تساوي 0 وهذا يعني أننا في الصف الأول ثم يقوم بالدخول إلى جملة الدوران الأخرى(الداخلية) التي سوف يكون فيها العداد j في البداية يساوي 0 وهذا يعني أننا في العمود الأول(أي نحن في الصف الأول والعمود الأول في البداية) ثم تزداد قيمة العداد j لننتقل إلى العمود الثاني وهكذا إلى أن نصل إلى العمود الثالث الذي رقمه 2 في الصف الأول فننتقل إلى الصف الثاني عندها تكون قيمة العداد i تساوي 1 وقيم العداد j تساوي 0 ويأخذ بالزيادة وهكذا ....إلى أن نصل إلى الصف الثاني والعمود الثالث أي قيمة العداد i بعد الخروج من الدوران 2 وقيمة العداد j تساوي 3 وهكذا تماما مع جملة الإدخال ولكن نستبدل جملة الطباعة بجملة الإدخال . cin>>a[i][j];

--- نستطيع أن نعرف مصفوفة ثنائية البعد ونقوم باعطاءها قيم أولية وذلك كالتالي:

datatype arrayname[][]={{v1,v2,v3,…},{w1,w2,w3,…},……..};

- **ملاحظات:**
- **يجوز أن نحدد حجم المصفوفة في هذه الحالة ونستطيع أيضا عدم تحديده .**
- **عند تحديد حجم المصفوفة يجب أن يكون عدد الصفوف يساوي أو يزيد عن عدد الصفوف المعطاة قيم أولية وعدد الأعمدة أيضا يجب أن يكون مساوي أو يزيد عن عدد الأعمدة المعطاة قيم أولية.**

- **إليك هذا المثال لتوضيح عملية تعريف مصفوفة ثنائية البعد وإعطائها قيم أولية مع رسم توضيحي:**

```
int arr[][]={{1,9,7,33},{55,12,90,0},{8,43,-10,56}};
```

تم تعريف مصفوفة ثنائية البعد عدد صفوفها 3 وعدد الأعمدة فيها 4 وإعطاءها قيم ابتدائية حيث أن ترتيب القيم فيها يكون كالتالي:

1	9	7	33
55	12	90	0
8	43	-10	56

- **ملاحظة هامة جدا:**

تسمى المصفوفة ثنائية البعد (مربعة) عندما يكون عدد الصفوف فيها يساوي عدد الأعمدة وعندها يجب معرفة وفهم الملاحظات التالية لمعرفة التعامل مع هذا النوع من المصفوفات: (على فرض أن لدينا مصفوفة حجمها 3*3)

1) **قيم موجودة في القطر الرئيسي للمصفوفة:**
- **العناصر التي تظهر بالون الأحمر الغامق تمثل عناصر القطر الرئيسي.**
- **إذا أردنا أن نتعامل مع القيم المخزنة في القطر الرئيسي فإننا نستخدم جملة if عندما يكون رقم الصف مساويا لرقم العمود.**

0,0	0,1	0,2
1,0	**1,1**	1,2
2,0	2,1	**2,2**

2) **القيم التي توجد أعلى القطر الرئيسي في المصفوفة:**
- **العناصر التي تظهر باللون الأحمر الغامق تمثل العناصر التي تقع أعلى القطر الرئيسي.**
- **إذا أردنا التعامل مع العناصر التي تقع أعلى القطر الرئيسي فإننا نستخدم جملة if عندما يكون رقم العمود اكبر من رقم الصف.**

0,0	**0,1**	**0,2**
1,0	1,1	**1,2**
2,0	2,1	2,2

3) القيم التي توجد أسفل القطر الرئيسي في المصفوفة:
- العناصر التي تظهر باللون الأحمر الغامق تمثل العناصر التي تقع أسفل القطر الرئيسي.
- إذا أردنا التعامل مع العناصر التي تقع أسفل القطر الرئيسي فإننا نستخدم جملة if عندما يكون رقم العمود اصغر من رقم الصف .

0,0	0,1	0,2
1,0	1,1	1,2
2,0	**2,1**	2,2

4) قيم موجودة في القطر العكسي للمصفوفة:
- العناصر التي تظهر باللون الأحمر الغامق تمثل عناصر القطر العكسي.
- إذا أردنا أن نتعامل مع القطر العكسي فإننا نستخدم جملة if عندما يكون مجموع رقم الصف ورقم العمود=(عدد الصفوف- 1).

0,0	0,1	**0,2**
1,0	**1,1**	1,2
2,0	2,1	2,2

5) القيم التي توجد أعلى القطر العكسي في المصفوفة:
- العناصر التي تظهر باللون الأحمر الغامق تمثل العناصر أعلى القطر العكسي.
- إذا أردنا أن نتعامل مع العناصر أعلى القطر العكسي فإننا نستخدم جملة if عندما يكون مجموع رقم الصف و رقم العمود اصغر من (عدد الصفوف-1).

0,0	**0,1**	0,2
1,0	1,1	1,2
2,0	2,1	2,2

6) القيم التي توجد أسفل القطر العكسي في المصفوفة:
- العناصر التي تظهر باللون الأحمر الغامق تمثل العناصر أسف القطر العكسي.
- إذا أردنا أن نتعامل مع العناصر أسفل القطر العكسي فإننا نستخدم جملة if عندما يكون مجموع رقم الصف و رقم العمود اكبر من (عدد الصفوف-1).

0,0	0,1	0,2
1,0	1,1	**1,2**
2,0	**2,1**	**2,2**

- **هذه مجموعة من الأمثلة على المصفوفات ثنائية البعد مع حلولها وشرحها:-**

1) اكتب برنامج يقوم بإدخال قيم صحيحة وتخزينها في مصفوفة ثنائية البعد حجمها 2*3 ثم يقوم بطباعة عناصر المصفوفة على شكل صفوف وأعمدة ثم يقوم بإيجاد مجموع القيم المخزنة في المصفوفة ومن ثم إيجاد الوسط الحسابي لها.

#include<iostream.h> void main(){ int x[3][2],i,j,sum=0; for(i=0;i<3;i++) for(j=0;j<2;j++) cin>>x[i][j]; for(i=0;i<3;i++) {for(j=0;j<2;j++) cout<<x[i][j]<<" "; cout<<endl;} for(i=0;i<3;i++) for(j=0;j<2;j++) sum+=x[i][j]; cout<<sum<<endl; cout<<sum/6.0<<endl;}	قمنا بتعريف مصفوفة ثنائية البعد بحجم 2*3 ومتغير اسمه sum وأعطيناه قيمة أولية 0 لأننا سوف نستخدمه للجمع لكي لا يؤثر على ناتج الجمع ثم استخدمنا جملة دوران متداخلة لكي نقوم بإدخال القيم إلى المصفوفة(صف صف) بعد ذلك استخدمنا جملة دوران متداخلة لكي تقوم بطباعة قيم المصفوفة ونلاحظ استخدام جملة cout<<endl; التي سوف تنفذ بعد طباعة كل صف وذلك لنقل المؤشر إلى السطر الجديد ومما يجعل طباعتنا بشكل صفوف وأعمدة(كما هو مطلوب في السؤال) واستخدمنا جملة دوران متداخلة للمرة الثالثة ولكن هذه المرة لجمع قيم المصفوفة وتخزين ناتج الجمع في المتغير الذي خصصناه لهذه العملية sum ثم قمنا بطباعة ناتج الجمع وبعدها قمنا بطباعة ناتج المتوسط الحسابي بعد قسمة المجموع على عدد العناصر في المصفوفة الذي يساوي 6 قيم لان حجم المصفوفة 2*3 .

2) اكتب برنامج يقوم بإدخال قيم صحيحة إلى مصفوفة حجمها 3*3 ثم يقوم بطباعة عناصر القطر الرئيسي وإيجاد مجموع القيم المخزنة به ومتوسطها الحسابي

#include<iostream.h> void main(){ int ar[3][3],i,j,sum=0; for(i=0;i<3;i++) for(j=0;j<3;j++) cin>>ar[i][j]; for(i=0;i<3;i++) for(j=0;j<3;j++) if(i==j) {cout<<ar[i][j]<<endl; sum+=ar[i][j];} cout<<"sum:"<<sum<<endl; cout<<"avg:"<<sum/3<<endl;}	عرفنا مصفوفة ثنائية البعد بحجم 3*3 ومتغير sum قيمته الأولية 0 لكي نستخدمه لعملية الجمع وعملية المتوسط الحسابي وبعدها استخدمنا جملة دوران متداخلة (لأننا نتعامل مع مصفوفة ثنائية البعد) لإدخال القيم الصحيحة إلى المصفوفة واستخدمنا جملة دوران متداخلة أخرى تقارن عدد الصفوف بعدد الأعمدة فإذا كانا متساويان أي أن ناتج عملية المقارنة true فهذا يعني أننا وصلنا إلى قيمة من قيم القطر الرئيسي فيقوم بطباعته ومن ثم يقوم بجمع هذه القيمة ويخزنها في المتغير sum وهكذا مع جميع القيم في القطر الرئيسي وبعدها بعد الخروج من الدوران يقوم بطباعة ناتج الجمع وناتج المتوسط الحسابي وذلك بقسمة القيمة المخزنة في sum على عدد العناصر الموجودة في القطر الرئيسي التي عددها 3 .

3) اكتب برنامج يقوم بإدخال أحرف إلى مصفوفة حجمها 4*4 ثم يقوم بطباعة العناصر الموجودة في أعلى القطر الرئيسي والعناصر الموجودة أسفل القطر العكسي.

#include<iostream.h> void main(){ char a[4][4];int i,j; for(i=0;i<4;i++) for(j=0;j<4;j++) cin>>a[i][j]; for(i=0;i<4;i++) for(j=0;j<4;j++) if(j>i) cout<<a[i][j]<<endl; for(i=0;i<4;i++) for(j=0;j<4;j++) if((i+j)>3) cout<<a[i][j]<<endl;}	عرفنا مصفوفة ثنائية البعد حجمها 4*4 من نوع char لان القيم المدخلة إليها هي أحرف ومن ثم استخدمنا جملة دوران متداخلة باستخدام عدادين مختلفين i للدوران الأول(الرئيسي) و j للدوران الداخلي تقوم بإدخال الأحرف إلى المصفوفة ثم استخدمنا جملة دوران متداخلة أخرى لطباعة العناصر الموجودة أعلى القطر الرئيسي وذلك باستخدام الشرط if(j>i) الذي يعني عندما يكون رقم العمود اكبر من رقم الصف واستخدمنا جملة دوران متداخلة أخرى لكي يقوم بطباعة عناصر التي تقع أسفل القطر العكسي وذلك باستخدام الشرط if((i+j)>3) الذي يعني عندما يكون مجموع رقم الصف ورقم العمود اكبر من عدد الصفوف – 1 أي في مثالنا 3=1-4 .

4) اكتب برنامج يقوم بإدخال قيم صحيحة إلى مصفوفة حجمها 3*3 ثم يقوم بإيجاد ناتج جمع كل صف من صفوفها ومجموع القيم الموجودة في الصف الثاني ومجموع القيم الموجودة في العمود الأول.

#include<iostream.h> void main(){ int x[3][3],i,j,rowsum=0,colsum=0,sum; for(i=0;i<3;i++) for(j=0;j<3;j++) cin>>x[i][j]; for(i=0;i<3;i++) {sum=0; for(j=0;j<3;j++) sum=sum+x[i][j]; cout<<"Sum:"<<sum<<endl;} for(i=0;i<3;i++) {rowsum+=x[1][i]; colsum+=x[i][0];} cout<<"RowSum:"<<rowsum<<endl; cout<<"ColSum:"<<colsum<<endl;}	عرفنا مصفوفة ثنائية البعد حجمها 3*3 ومتغير اسمه rowsum قيمته الأولية 0 لكي يساعدنا في جمع القيم الموجودة في الصف الثاني ومتغير أخر colsum قيمته الأولية أيضا 0 لكي يساعدنا في جمع القيم في العمود الأول ومتغير اسمه sum لم يعطى قيمة أولية وقمنا بتعريفه لكي يساعدنا في جمع القيم الموجودة في كل صف على حدا ثم استخدمنا جملة دوران متداخلة لكي نقوم بإدخال القيم إلى المصفوفة ومن ثم قمنا باستخدام جملة دوران متداخلة أخرى لكي نقوم بحساب مجموع القيم في كل صف ولكن أعطينا المتغير sum قيمة أولية 0 بين جملة الدوران الرئيسية وجملة الدوران الداخلية لكي يقوم بتصفير المتغير sum بعد جمع كل صف لكي لا يؤثر على مجموع الصفوف الأخرى وبعد الانتهاء من جمع كل صف نقوم بطباعة ناتج الجمع ثم استخدمنا جملة دوران غير متداخلة لكي تقوم بجمع القيم في العمود الأول والقيم في الصف الثاني واستخدمناها غير متداخلة لأنه في الصف الثاني الذي رقمه 1 فقط الذي يتغير رقم العمود وفي العمود الأول الذي رقمه 0 لا يتغير إلا رقم الصف ثم قمنا بطباعة ناتج الجمع.

5) اكتب برنامج يقوم بإدخال قيم صحيحة إلى مصفوفة حجمها 3*3 ثم يقوم بتبديل الصف الأول مع الصف الأخير.

#include<iostream.h> void main(){ int amr[3][3],i,j,t; for(i=0;i<3;i++) for(j=0;j<3;j++) cin>>amr[i][j]; for(i=0;i<3;i++) {t=amr[0][i]; amr[0][i]=amr[2][i]; amr[2][i]=t;}	عرفنا مصفوفة ثنائية البعد حجمها 3*3 ومتغير اسمه t لكي يساعدنا في عملية التبديل ثم استخدمنا جملة دوران متداخلة لإدخال القيم إلى المصفوفة وثم استخدمنا جملة دوران أخرى(غير متداخلة) لان رقم الصفوف التي نريد تبديلها معروفة لدينا ثم قمنا بتخزين القيمة الأولى في الصف الأول في المتغير t ثم خزنا القيمة الأولى في الصف الأخير في موقع القيمة الأولى في الصف الأول ثم خزنا القيمة الأولى في الصف الأول(الموجودة في المتغير t)في موقع القيمة الأولى في الصف الأخير وهكذا للقيمة الثانية والثالثة وذلك بسبب استخدامنا لجملة الدوران التي تساعدنا للانتقال لباقي القيم وذلك باستخدام عدادها i.

6) اكتب برنامج يقوم بإدخال قيم صحيحة إلى مصفوفة حجمها 3*3 وثم يقوم بإيجاد اكبر قيمة واصغر قيمة في المصفوفة.

#include<iostream.h> void main(){ int cs[3][3],max,min; for(int i=0;i<3;i++) for(int j=0;j<3;j++) cin>>cs[i][j]; max=min=cs[0][0]; for(i=0;i<3;i++) for(int j=0;j<3;j++) {if(cs[i][j]>max) max=cs[i][j]; if(cs[i][j]<min) min=cs[i][j];} cout<<"Max:"<<max<<endl<<"Min:"<<min; cout<<endl;}	عرفنا مصفوفة حجمها 3*3 ومتغيرين الأول اسمه max وذلك لنخزن القيمة الأكبر فيه والمتغير الأخر اسمه min وذلك لتخزين القيمة الأصغر فيه ثم استخدمنا جملة دوران متداخلة لإدخال القيم إلى المصفوفة وبعدها قمنا بفرض أن القيمة الأولى في المصفوفة بأنها القيمة الأكبر والقيمة الأصغر وذلك لمقارنتها مع باقي قيم المصفوفة ثم استخدمنا جملة دوران متداخلة أخرى لكي نقوم بمقارنة جميع العناصر مع القيمة المفترضة فإذا تم العثور على قيمة اكبر من القيمة المفترضة سيتم تخزينها في المتغير max وإذا تم العثور على قيمة اصغر من القيمة المفترضة سيتم تخزينها في المتغير min ثم نقوم بطباعة الناتج. لاحظ إننا قمنا بتعريف المتغير i و j داخل جملة الدوران المتداخلة الأولى وهذا يجوز بشرط أن لا تقوم بتعريف المتغير i مرة أخرى في جمل الدوران اللاحقة بعكس المتغير j الذي يجب تعريفه في كل جملة دوران لاحقة وذلك لأنه ضمن جملة الدوران الداخلية.

7) اكتب برنامج يقوم بإدخال قيم صحيحة إلى مصفوفة حجمها 3*3 ثم يقوم بتغير قيمة العنصر الموجود في منتصف المصفوفة ثم طباعتها بعد التغير.

#include<iostream.h> void main(){ int c[3][3],i,j; for(i=0;i<3;i++) for(j=0;j<3;j++) cin>>c[i][j]; for(i=0;i<3;i++) for(j=0;j<3;j++) if(i==2 && j==2) cout<<"Enter a Value:"<<endl; cin>>c[1][1]; for(i=0;i<3;i++) for(j=0;j<3;j++) cout<<c[i][j]<<" ";}	عرفنا مصفوفة ثنائية البعد حجمها 3*3 ثم استخدمنا جملة دوران متداخلة لإدخال القيم إلى المصفوفة ثم استخدمنا جملة دوران أخرى أيضا متداخلة وذلك للوصول إلى العنصر الذي يقع في منتصف المصفوفة الذي يكون في مصفوفتنا العنصر الموجود في العمود الثاني الذي رقمه 1 والعمود الثاني الذي رقمه 1 فعندما نصل إليه نقوم بإدخال القيمة لكي نقوم بتبديلها مع القيمة الموجودة في منتصف المصفوفة وبعدها استخدمنا جملة دوران متداخلة لكي نقوم بطباعة عناصر المصفوفة بشكل افقي وذلك بعد تغيير القيمة.

8) اكتب برنـامج يقوم بتخزين جداول الضرب للإعداد 2 و5 و 9 في مصفوفة بحيث يكون كل عدد جدوله في صف.

قمنا بتعريف مصفوفة ثنائية البعد حجمها 10*3 وذلك لان لدينا ثلاث أعداد وكل عدد جدوله مكون من 10 قيم من دون ضرب العدد بالرقم 10 وابتداء من ضرب العدد ب 0 وبعدها استخدمنا جملة دوران متداخلة وذلك لتخزين جداول الضرب لكل عدد ابتداء من العدد 2 في الصف الأول حتى العدد 9 في الصف الأخير(الثالث) وذلك بضرب العدد الذي نريد جدوله مع العداد j ثم نخزن القيم في صفوف المصفوفة.	<pre>#include<iostream.h> void main(){ int x[3][10],i,j; for(i=0;i<3;i++) for(j=0;j<10;j++) {x[0][j]=5*j; x[1][j]=2*j; x[2][j]=9*j;} }</pre>

9) اكتب برنـامج يقوم بإدخال قيم إلى مصفوفة حجمها 2*2 ثم يقوم بنقل قيم المصفوفة إلى مصفوفة أخرى وذلك بشرط أن تكون القيمة اكبر من العدد 5 فإذا كانت غير ذلك نقوم بإدخال قيمة اكبر من العدد 5 بدل منها.

عرفنا مصفوفتين الأولى حجمها 2*2 ثنائية البعد لكي نقوم بإدخال القيم إليها والثانية حجمها أيضا 2*2 لكي نقوم بنقل القيم الأكبر من العدد 5 أليها ثم استخدمنا جملة دوران متداخلة لكي نقوم بإدخال القيم إلى المصفوفة الأولى amvb1 وبعدها استخدمنا جملة دوران أخرى متداخلة أيضا وذلك لنقل القيم الأكبر من العدد 5 الموجودة في المصفوفة amvb1 إلى المصفوفة amvb2 فإذا كانت هناك قيم في المصفوفة الأولى amvb1 اصغر من العدد 5 أو تساويه فعند العثور عليه سوف يتم إدخال قيمة اكبر من العدد 5 بدلا منها لكي تخزن القيمة الجديدة في المصفوفة الثانية amvb2 وبنفس الموقع الذي يحتوي على القيمة الأصفر من العدد 5 أو تساويه ولكن في المصفوفة الثانية amvb2 .	<pre>#include<iostream.h> void main(){ int amvb1[2][2],amvb2[2][2],i,j,v; for(i=0;i<2;i++) for(j=0;j<2;j++) cin>>amvb1[i][j]; for(i=0;i<2;i++) for(j=0;j<2;j++) if(amvb1[i][j]>5) amvb2[i][j]=amvb1[i][j]; else {cout<<"Enter a value bigger than 5:"<<endl; cin>>v;} }</pre>

10) اكتب برنامج يقوم بإدخال قيم إلى مصفوفتين حجم كل منهما 3*3 ثم يقوم بنقل قيم القطر الرئيسي في المصفوفة الأولى إلى القطر العكسي في المصفوفة الثانية بحيث يصبح القطر العكسي للمصفوفة الثانية هو القطر الرئيسي للمصفوفة الأولى.

<table>
<tr>
<td>

```
#include<iostream.h>
void main(){
int a[3][3],b[3][3],i,j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
cin>>a[i][j];
for(i=0;i<3;i++)
for(j=0;j<3;j++)
cin>>b[i][j];
int k=2;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
if(i==j)
{b[i][k]=a[i][j];
k--;}
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
cout<<b[i][j]<<" ";
cout<<endl;}
}
```

</td>
<td>عرفنا مصفوفتين الأولى حجمها 3*3 اسمها a وقمنا بإدخال القيم إليها من خلال جملة الدوران الأولى المتداخلة والثانية حجمها أيضا 3*3 اسمها b قمنا بإدخال القيم إليها من خلال جملة دوران أخرى أيضا متداخلة بعد ذلك عرفنا متغير اسمه k له قيمة أولية 2 وذلك لمساعدتنا في عملية النقل ثم استخدمنا جملة دوران متداخلة أخرى لكي نقوم بنقل قيم القطر الرئيسي في المصفوفة الأولى a إلى القطر العكسي في المصفوفة الثانية b وذلك باستخدام جملة if فإذا كان ناتجها true فهذا يعني أننا وصلنا لعنصر من عناصر القطر الرئيسي في المصفوفة الأولى فنقوم بنقله من المصفوفة الأولى إلى الصف الأول العمود الثالث الذي أشار إليه المتغير k لان قيمته الأولية 2 ثم نقوم بتنقيص قيمة المتغير k بمقدار واحد لكي يتم النقل في المرة اللاحقة في الصف الثاني والعمود الثاني وبعدها أيضا لكي نقوم بالنقل إلى الصف الثالث العمود الأول وبهذه الطريقة نكون قد نقلنا القيم من القطر الرئيسي في المصفوفة الأولى a إلى القطر العكسي في المصفوفة الثانية b ثم استخدمنا جملة دوران متداخلة أخيرة لكي نقوم بطباعة قيم المصفوفة الثانية b بطريقة الصفوف والأعمدة بعد النقل .</td>
</tr>
</table>

البرامج الفرعية (Functions)

- هناك نوعان من البرامج الفرعية:

1) برامج فرعية جاهزة: وهي التي تكون معرفة مسبقا من قبل اللغة وما على المبرمج إلا أن يستدعيها مباشرة.
2) برامج فرعية نحن نقوم بتعريفها: وهي التي يقوم المبرمج بإنشائها وكتابة مجموعة من الجمل لتقوم بوظيفة معينة.

1) برامج فرعية جاهزة \\

- يهمنا فقط دراسة جزء منها والأكثر استخداما
- إليك هذا الجدول بأهم هذه الدوال الجاهزة مع وصف وأمثلة توضيحية لكل واحدة:

الدالة	الوصف	أمثلة توضيحية
ceil(x)	تقوم بتقريب قيمة x إلى عدد صحيح اكبر من x	ceil(0.2)→ 1 ceil(-5.1)→ -5 ceil(9)→ 9
floor(x)	تقوم بتقريب قيمة x إلى عدد صحيح اصغر من x	floor(2.9)→ 2 floor(-1,2)→ -2 floor(4)→ 4
sin(x)	لإيجاد جيب الزاوية x	sin(0.0)→ 0
cos(x)	لإيجاد جيب التمام للزاوية x	cos(0.0)→ 1
tan(x)	لإيجاد ظل الزاوية x	tan(0.0)→ 0
sqrt(x)	لإيجاد الجذر التربيعي للعدد x	sqrt(25)→ 5 sqrt(100)→ 10
pow(x,y)	لإيجاد ناتج رفع قيمة x إلى القوة y	pow(6,1)→ 6 pow(2,0)→ 1
fmod(x,y)	لإيجاد باقي قسمة x على y	fmod(7,3)→ 1 fmod(4.5,2.2)→ 0.1
fabs(x)	لإيجاد القيمة المطلقة ل x	fabs(-4.4)→ 4.4 fabs(-6)→ 6 fabs(0)→ 0

ملاحظات:

- نقوم باستدعاء أي دالة من الدوال السابقة بالطريقة التالية:

function_name(list of arguments);

حيث أن: function_name: اسم الدالة التي نريد استخدامها.
list of argument: القيم التي نريد إرسالها للدالة فإذا كان هناك أكثر من قيمة يتم الفصل بينهما بفواصل

- لا يمكننا استخدام أي دالة من الدوال السابقة في برنامجنا إلا إذا قمنا بكتابة #include<math.h> في بداية البرنامج قبل main( )
- عند استخدام الدالة sin(x) أو cos(x) أو tan(x) يجب أن تكون الزاوية x معطاة بالتقدير الدائري.
- عند استخدام الدالة fmod أو fabs يمكن أن نرسل قيم صحيحة أو قيم حقيقية(غير صحيحة).

**2) برامج فرعية نحن نقوم بتعريفها **

- **عليك أن تتفهم الأمور التالية لتكون قادر على كتابة أي برنـامج فرعي (دالة):-**
- **مكان كتابة البرنـامج الفرعي.**
- **كيفية كتابة والتعامل مع الجمل التي تقع ضمن البرنـامج الفرعي.**
- **كيفية استدعاء البرنـامج الفرعي.**

أ) مكان كتابة البرنـامج الفرعي:

- **يكتب البرنـامج الفرعي قبل ال main ويكتب أيضا بعد أل main ولكن يجب الانتباه فإذا كان البرنـامج مكتوب بعد أل main فانه يحتاج إلى كتابة نموذج لـهذا البرنـامج الفرعي قبل أل main وهو ما يسمى ب(prototype) أما إذا كان قبل main فانه لا يحتاج إلى ذلك.**

- **ولكن كيف نقوم بكتابة prototype؟؟؟؟؟**

-- الشكل العام لكتابته: function_name(datatype variables_name);

- function_name: وهو اسم من اختيارك ويفضل أن يعبر عن وظيفة الدالة ويخضع اسم الدالة إلى شروط تسمية المتغيرات.

- datatype: نوع البيانات للمتغير الذي سوف تخزن به القيمة التي يستقبلـها هذا البرنـامج الفرعي أو الدالة.

- variables_name: اسم المتغير الذي تستخدمه الدالة لكي تستقبل القيم وإذا كان لدينا أكثر من متغير نفصل بين كل متغيرين بفاصلة ويمكن أن تستقبل الدالة عدد من المتغيرات ولكل متغير نوع بيانـات خاص به. وقد تكون الدالة لا تستقبل أي متغير.

- (انتبه:يعد كتابة اسم المتغير في prototype اختياري).

- لا تنسى الفاصلة المنقوطة بعد كتابتك لل prototype.

ب) كيفية كتابة والتعامل مع الجمل التي تقع ضمن البرنـامج الفرعي:

- **لك الحرية في اختيار الجمل التي تريدها داخل الدالة من جمل طباعة وجمل إدخال وجمل تعريف وتعيين وغيرها**
- **هناك جملة يجب استخدامها عندما ترجع الدالة قيمة وهي return وتستخدم لإرجاع قيمة .**

- **يجب الانتباه عند تعريف المتغيرات داخل الدالة:**

-- فعند تعريف أي متغير داخل الدالة يسمى متغير محلي (local) فهذا المتغير هو ملك لـهذه الدالة ولا يجوز لل main ولا لأي دالة أخرى استخدامه مباشرة.

ج) كيفية استدعاء البرنـامج الفرعي:

- **يستدعى البرنـامج الفرعي (الدالة) من أل main ويمكن استدعاءه من دالة أخرى ولكلا الحالتين الطريقة التالية:**

-- الشكل العام لاستدعاء الدالة: function_name(variabeles_name);

- **variabeles_name: اسم المتغير أو المتغيرات(مفصولة بفواصل) التي نقوم بإرسالـها للدالة ويجب أن يكون المتغير معرف ضمن المجال الذي تستدعي به الدالة فإذا كان استدعاء الدالة من أل main فيجب تعريف المتغير ضمن أل main أما إذا كان استدعاءه من دالة أخرى فيجب أن يكون المتغير معرف ضمن هذه الدالة**
- **يمكن أن تكون الدالة لا تستقبل أي قيمة(لا نرسل أي قيمة للدالة أي أننا نترك الأقواس فارغة أو نقوم بكتابة بين القوسين كلمة void .**
- **لا تنسى الفاصلة المنقوطة عند استدعاءك لأي دالة.**

- **أنواع البرامج الفرعية من حيث إرجاع القيمة:**

1) دوال ترجع قيمة :

- **عندما ترجع الدالة قيمة يمكن أن تستدعى الدالة بشكل منفرد ويجوز أن تكون ضمن جمل طباعة أو أن تكون ضمن جملة تعيين ولا يجوز ذلك لو انه لا يرجع قيمة إلا استدعاءه بشكل منفرد.**

- **عندما تكون الدالة ترجع قيمة فانه يجب تحديد نوع البيانات التي سوف ترجعها الدالة قبل اسم الدالة عند كتابتها(تعريفها) وقبل اسم الدالة عندما نكتب لها prototype ولا نحدده عندما نقوم باستدعائها.**

- **يمكن أن تكون أنواع البيانات التي ترجعها الدالة int,float,double,bool,char............**

- **لكي تحصل على نتائج صحيحة ومتوقعة عندما ترجع الدالة قيم وقمنا باستدعاء الدالة ضمن جملة تعيين فيجب أن يكون المتغير الذي سوف تخزن به القيمة التي ترجعها هذه الدالة من نفس نوع البيانات المرجعة.**

- **عندما تكون الدالة ترجع قيمة فيجب أن تحتوي الدالة على جملة return**

2) دوال لا ترجع قيمة:

- **عندما تكون الدالة لا ترجع قيم فلا يجوز أن تستدعى الدالة إلا بشكل منفرد(بدون استخدام أي جملة أخرى)**

- **عندما تكون الدالة لا ترجع قيم يجب كتابة void قبل اسم الدالة عند كتابتها(تعريفها) وقبل اسم الدالة عندما نكتب prototype لها ولا نكتبها عند استدعاءها.**

- **لا تستخدم جملة return عندما تكون الدالة لا ترجع قيم.**

← هناك طريقتين لاستدعاء البرامج الفرعية:

-- عند إرسال القيم لبرامج الفرعية من خلال المتغيرات هناك حالتين:

1) إرسال مرجع المتغير (عنوانه). Call by reference
- **يجب وضع هذه الإشارة قبل اسم المتغير لكي نحصل على موقع أو مرجع المتغير &.**

- **عندما نستخدم هذه الطريقة فان التغيير على المتغير يحفظ بعد استدعاء الدالة.**

2) إرسال صورة عن القيمة. Call by value
- **عندما نستخدم هذه الطريقة فان أي تغيير على المتغير لا يحفظ بعد استدعاء الدالة.**

:((Over Loading Functions)) ←

ـ يجوز تسمية أكثر من دالة بنفس الاسم ولكن بشرط أن تختلف قائمة المتغيرات المرسلة إليها من حيث العدد والترتيب والنوع وهذا يسمى (over loading).

ـ يمكننا تحديد الدالة التي نريد استدعاءها من خلال عدد المتغيرات وترتيبها ولا يهم إن اختلف نوع البيانات المرجعة فلا يجوز أن نقوم بتسمية دالتين بنفس الاسم وبنفس عدد المتغيرات المستقبلة وبنفس الترتيب ولكن هناك اختلاف في أنوع البيانات المرجعة.

:((Recursive Functions)) ←

- هناك نوع من الدوال يسمى Recursive Functions : أي دالة تقوم باستدعاء نفسها ضمن شرط معين.
- عادة تستخدم مع هذا النوع من البرامج الفرعية جمل IF للتحكم بتوقف استدعاء الدالة لنفسها.
- سيتضح لك هذا النوع من الدوال في الأمثلة اللاحقة.

:((Template Functions)) ←

- عندما نريد أن نقوم بكتابة دالة تقوم بعمل معين وتستقبل قيم من أنواع مختلفة أي أنها قد تستقبل متغيرات من نوع int وقد تحتاج لاستقبال متغيرات من نوع char وغيرها من أنواع البيانات فهذا يقودنا إلى عمل عدة برامج فرعية ولكل واحد نوع من البيانات ولكن لغة c++ قدمت الحل باستخدام template functions الذي يوفر الوقت والجهد بدلا من كتابة عدد من الدوال تقوم بنفس العمل سنقوم فقط بكتابة دالة واحدة لجميع أنواع البيانات المختلفة.

- كيفية استخدام template functions ؟؟؟

ــ إذا قمنا بكتابة الدالة قبل أل main يجب كتابة قبل تعريف الدالة template <class t> ونقوم بكتابة t بدل نوع البيانات المرجعة ونوع البيانات للمتغيرات التي نستقبلها.

ــ إذا كتبنا الدالة بعد أل main فانه يجب كتابة template<class t> مرتين:
الأولى: عند تعريفها(كتابتها) .
الثانية: قبل prototype للدالة.

- كيف يعمل template function ؟؟؟

ـ عند استدعاء دالة قمنا مسبقا بتعريف template function لها فان نوع البيانات لمتغيرات الدالة سوف يتغير حسب القيم المرسلة فلو تم إرسال قيم من نوع int سوف يتغير t ويصبح int وإذا تم إرسال قيم من نوع char تصبح t char وهكذا....

- t : رمز نحن قمنا باختياره يمكنك اختيار أي رمز آخر بدل منه.

:((Based Arrays To Functions)) ←

-- يمكننا تعريف دوال تقوم بوظائف خاصة بالمصفوفات كإدخال قيم لمصفوفة من خلال دالة أو طباعة قيم مصفوفة وغير ذلك من أمور تحتاجها المصفوفة ((كيف؟؟؟)):

- هناك فقط اختلافين بسيطين عن الطرق التي ذكرناها سابقا (ففي حال كانت المصفوفة أحادية البعد) :
الاختلاف الأول: عند كتابة prototype فعند كتابته نقوم باستقبال المصفوفة بحيث نكتب نوع البيانات للمصفوفة وبعدها اسم مصفوفة افتراضي(يجوز أن يكون اسمها نفس اسم المصفوفة الحقيقية وبعد كتابة اسم المصفوفة في prototype اختياري) ثم نضع هذه الأقواس المربعة [] .
الاختلاف الثاني: عند استدعاء الدالة نقوم فقط بإرسال اسم المصفوفة وعند تعريف الدالة أو كتابتها نقوم بكتابة أل prototype نفسه ولكن نضع اسم للمصفوفة افتراضي لكي نقوم باستقبال المصفوفة الحقيقية من خلاله.

- (أما في حال كانت المصفوفة ثنائية البعد) :
نقوم بتحديد عدد الصفوف وعدد الأعمدة قبل أل main وذلك بتخزين عدد الصفوف بثابت معين وتخزين عدد الأعمدة بثابت معين وعند كتابة ألprototype للدالة يجب أن يوضع نوع البيانات للمصفوفة التي نريد استقبالها ثم [عدد الأعمدة من خلال اسم الثابت الذي خزنا به عدد الأعمدة][ويجوز أن نحدد عدد الصفوف من خلال الثابت الذي خزنا به الصفوف] ولكن تحديد الصفوف يكون اختياري. فمثلا لو قمنا بتخزين عدد الصفوف والأعمدة قبل أل main بالشكل التالي:
const int row=3; const int col=3; فانه سيكون شكل أل prototype كالتالي:
(datatype function_name[row][col]); أو (datatypefunction_name [][col]);
ويجب عمل ذلك أيضا عند كتابة الدالة (تعريفها) ولكن نضيف اسم افتراضي لكي نستقبل المصفوفة الحقيقية من خلاله (يجوز أن يكون الاسم الافتراضي الذي نستقبل به المصفوفة نفس اسم المصفوفة الحقيقية)

- عند الاستدعاء لا يوجد اختلاف عن طريقة استدعاء المصفوفة أحادية البعد (إرسال اسم المصفوفة فقط).
- في أل main عند تعريف المصفوفة لا نحدد عدد الصفوف والأعمدة للمصفوفة من خلال الأرقام بل نستخدم الثوابت التي خزنا بها عدد الأعمدة والصفوف كالتالي: int array[row][col];
- تعتبر طرية استقبال المصفوفات في الدوال call by reference أي يجب أن نتذكر أي تغيير أو تعديل على قيم المصفوفة يبقى ويحفظ.

-- ملاحظات هامة:
1) - يمكن أن يكون شكل أل prototype كالتالي:
datatype function_name(datatype=value1,datatype=value2,.........);
في هذه الحالة يجب معرفة الأمور التالية:
* لو فرضنا أن لدينا prototype كالسابق بقيمتين فقط :
- فعند استدعاء الدالة وإرسال قيمتين سوف تستخدم الدالة القيم التي قمنا بإرسالها (الجديدة).
- عند استدعاء الدالة وإرسال قيمة واحدة فقط فإن الدالة تستخدمها بدل من القيمة الأولى في أل prototype وتستخدم القيمة الثانية الموجودة فيها مسبقا.
- عند استدعاء الدالة وعدم إرسال أي قيمة سوف نقوم باستخدام القيمتين في أل prototype الموجودات مسبقا.
2) - عليك الانتباه من استخدام static مع المتغيرات:
فعند استخدامك ل static مع أي متغير فهذا يعني أي تغير على قيمة المتغير سوف يحفظ ويخزن به
3) – يكون نوع المتغيرات global إذا كان غير موجود بأي دالة ولا حتى بال main وفي هذه الحالة تستطيع جميع الدوال وال main باستخدامه والتغيير عليه (سيتوضح لك ذلك أكثر بالأمثلة اللاحقة).

* هذه مجموعة من الأمثلة على البرامج الفرعية مع حلولها وشرحها:-

1) اكتب دالة تقوم بإيجاد العدد الأكبر من ثلاث أعداد صحيحة يتم إدخالها عن طريق لوحة المفاتيح.

قمنا في البداية قبل أل main بكتابة أل prototype للدالة اسمها max وترجع قيمة من نوع int ثم في ألmain قمنا بتعريف متغير اسمه max_value لنخزن به القيمة التي سوف ترجعها الدالة وهي القيمة الأكبر ثم استدعينا الدالة وخزنا القيمة التي سوف تقوم بإرجاعها في المتغير الذي خصصناه لذلك ثم قمنا بطباعة قيمة المتغير الذي يحتوي على القيمة الأكبر التي ترجعها الدالة ثم قمنا بعد أل main قمنا بكتابة الدالة وعرفنا بداخلها أربع متغيرات n1,n2,n3 لإدخال القيم من لوحة المفاتيح ومتغير max لنخزن به العدد الأكبر ثم أجرينا عمليات المقارنة اللازمة لإيجاد العدد الأكبر وخزنا العدد الأكبر في المتغير max ثم استخدمنا جملة return لإرجاع القيمة الأكبر.	`#include<iostream.h>` `int max();` `void main(){` `int max_value;` `max_value=max();` `cout<<max_value<<endl;}` `int max(){` `int n1,n2,n3,max;` `cin>>n1>>n2>>n3;` `max=n1;` `if(n2>max)` `max=n2;` `if(n3>max)` `max=n3;` `return max;}`

2) اكتب دالة تقوم بإيجاد مربع لأي عدد صحيح نقوم بإدخاله .

قمنا بتعريف الدالة قبل ألmain فلذلك لم نحتاج إلى كتابة prototype وهذه الدالة لا ترجع قيمة بل تقوم بطباعة الناتج على الشاشة مباشرة من خلال جملة الطباعة cout التي تقوم بطباعة مربع العدد. تقوم الدالة باستخدام متغير x وهو الذي يحمل القيمة التي يبعثها main لهذه الدالة وهذا المتغير يعبر عن قيمة المتغير n ثم قمنا بإدخال عدد من لوحة المفاتيح وأرسلناها للدالة واستدعيناها بشكل منفرد(لأنها لا ترجع أي قيمة)	`#include<iostream.h>` `void sq(int x){` `cout<<x*x<<endl;}` `void main(){` `int n;` `cin>>n;` `sq(n);}`

3) ما ناتج تنفيذ البرنامج التالي:

نلاحظ ناتج جملة الطباعة الأولى 14 لأنه تم إرسال قيمتين للدالة من خلال المتغيرين a و b وزيادة مقدار كل من القيمتين بمقدار واحد ثم جمع القيمتين وناتج جملة الطباعة الأخرى 9 5 وذلك بسبب إشارة & والتي تعني أننا نقوم بالتعديل على قيمة المتغير وحفظ هذا التغيير حتى بعد استدعاء الدالة فلو كانت الدالة تستقبل القيم من دون هذه الإشارة لكان ناتج تنفيذ الجملة الثانية 8 4 أي أن التغيير الذي أجريناه على المتغيرات لم يتم حفظه	`#include<iostream.h>` `int f(int &x,int &y){` `x++;` `y++;` `return x+y;}` `void main(){` `int a=4,b=8;` `cout<<f(a,b)<<endl;` `cout<<a<<" "<<b<<endl;}` --- 14 5 9

4) ما ناتج تنفيذ البرنامج التالي:

```
#include<iostream.h>
int amr(int m){
return m*m;}
double amr(double m){
return m+m;}
int amr(int m1,int m2){
return m1-m2;}
void main(){
cout<<amr(9,6)<<endl;
cout<<amr(8)<<endl;
cout<<amr(7.5)<<endl;}
```

```
3
64
15
```

نلاحظ ناتج تنفيذ جملة الطباعة الأولى 3 أي انه قام باستدعاء الدالة الثانية بالرغم أن لدينا ثلاثة دوال بنفس الاسم ولكن سيتم استدعاء الدالة المناسبة من خلال القيم المرسلة إليها. ففي البداية لأنه تم إرسال قيمتين تم استدعاء الدالة الأخيرة لأنها تستقبل قيمتين من نوع int , وناتج تنفيذ جملة الطباعة الأخرى 64 أي انه تم استدعاء الدالة التي تستقبل قيمة واحدة من نوع int التي تقوم بضرب القيمة بنفسها وناتج تنفيذ جملة الطباعة الأخيرة 15 لأنه سوف يتم استدعاء الدالة التي تستقبل قيمة من نوع double وتقوم بجمع القيمة لنفسها

5) ما ناتج تنفيذ البرنامج التالي:

```
#include<iostream.h>
template<class t>
t sub(t x, t y);
void main(){
cout<<sub(3,9)<<endl;
cout<<sub(3.5,0.5)<<endl;}
template<class t>
t sub(t x,t y){
t sub;
sub=x-y;
return sub;}
```

```
-6
3
```

ناتج جملة الطباعة الأولى 6- وناتج جملة الطباعة الثانية 3 مع أننا قمنا بتعريف دالة واحدة فقط ولكنها قادرة على طرح أي عددين مهما كان نوعاهما (أعدادا صحيحة أعداد حقيقية وغير ذلك) وذلك بسبب استخدام template الذي يسمح لنا باستخدام الدالة لأكثر من نوع من البيانات ولاحظ أننا قمنا بتعريف متغير في الدالة اسمه sub كان نوع البيانات له t أي انه سيتغير نوع بيانات هذا المتغير بناءا على نوع القيم التي سوف تستقبلها الدالة.

6) ما ناتج تنفيذ البرنامج التالي:

```
#include<iostream.h>
const int a=10;
int x=5;
void f();
void main(){
cout<<++x<<endl;
cout<<a<<endl;
f();
cout<<x<<endl;}
void f(){
cout<<x++<<endl;}
```

```
6
10
6
7
```

ناتج جملة الطباعة الأولى 6 وذلك لان الجملة تقوم بزيادة مقدار واحد على قيمة x ثم طباعتها وبما انه لا يوجد x في أل main مباشرة يتم استدعاء x العالمية global التي عرفناها قبل ألmain وناتج جملة الطباعة الثانية 10 لنفس السبب لجملة الطباعة الأولى ثم تم استدعاء الدالة التي تقوم بطباعة قيمة x ثم زيادتها بمقدار واحد وبما انه لا توجد x معرفة داخل الدالة (local variables) سيتم طباعة قيمة x العالمية global التي أصبحت قيمتها 6 من جملة الطباعة الأولى التي زادة قيمتها ثم بعد طباعتها زيادة قيمة x مقدار واحد ثم تم طباعة قيمة x من خلال جملة الطباعة الأخيرة في main التي ستقوم بطباعة 7 لان قيمة x أصبحت 7 بعد استدعاء الدالة التي زادة على مقدار هذا المتغير مقدار واحد.

7) ما ناتج تنفيذ البرنامج التالي:

#include<iostream.h> int a(int x=5,int y=10); void main(){ cout<<a(2,3)<<endl; cout<<a(5)<<endl; cout<<a()<<endl;} int a(int x,int y){ return x+y;} ——— 5 15 15	نلاحظ انه تم كتابة prototype قبل الmain للدالة وقمنا بإعطاء قيم لمتغيرات الدالة فعند تنفيذ البرنامج سوف يكون ناتج تنفيذ جملة الطباعة الأولى 5 وذلك لأننا أرسلنا قيمتين للدالة الأولى 2 والثانية 3 وتم جمعهما من خلال الدالة أما عند تنفيذ جملة الطباعة الثانية سيكون الناتج 15 لأنه تم إرسال قيمة واحدة فقط للدالة فتم تبديلها بالقيمة الأولى التي تم تخزينها في المتغير x عند كتابة prototype والقيمة المخزنة في المتغير y سوف تبقى وعند جمع 5 مع 10 سوف يكون الناتج 15 أما ناتج جملة الطباعة الأخيرة لم يتم إرسال أي قيمة للدالة فلذلك سوف تستخدم الدالة القيم الأولية المخزنة بها 5 و10 وعند جمعهما سيكون الناتج 15

8) ما ناتج تنفيذ البرنامج التالي:

#include<iostream.h> int x=2; void g(int x){ cout<<x<<endl;} void main(){ int x=0; g(x); g(::x);} ——— 0 2	عند استدعاء الدالة الأولى سيتم طباعة القيمة 0 لان قيمة x التي تم إرسالها 0 القريبة(local) أما عند استدعاء الدالة الثانية سيتم طباعة القيمة 2 لان قيمة x التي تم إرسالها 2 البعيدة (global) وذلك بسبب استخدامنا :: التي تأتي بقيمة المتغير البعيد

9) ما ناتج تنفيذ البرنامج التالي:

#include<iostream.h> int stat(){ static int x=6; x+=9; return x;} void main(){ cout<<stat()<<endl; cout<<stat()<<endl;} ——— 15 24	ناتج جملة الطباعة الأولى 15 وذلك لأننا استدعينا دالة عرفت متغير سبقته كلمة static وأعطيناه قيمة أولية 6 ثم جمعنا هذه القيمة مع العدد 9 أما ناتج جملة الطباعة الثانية ستكون 24 وذلك لان القيمة المخزنة في المتغير x هي 15 أي أخر قيمة لهذا المتغير وذلك بسبب استخدام static التي تعمل على تخزين أخر قيمة للمتغير فلذلك القيمة المخزنة في x بعد استدعاء الدالة في المرة الأولى هي 15 ثم قمنا بجمع العدد 9 له فأصبحت قيمة المتغير x 24.

10) ما ناتج تنفيذ البرنامج التالي:

`#include<iostream.h>` `int x=6.5;` `void print(int x){` `x++;` `cout<<x<<endl;}` `void main(){` `{int x=4;` `print(x);}` `cout<<x<<endl;` `print(x);}` ——— 5 6 7	ناتج استدعاء الدالة في المرة الأولى سيكون الناتج 5 لان قيمة المتغير x المرسلة للدالة 4 والدالة تقوم بزيادة مقداره بمقدار واحد ثم طباعة قيمته ثم بعدها سيكون ناتج تنفيذ جملة الطباعة في ألmain 6 لان قيمة x هي 6.5 وبما أن نوع البيانات للمتغير int x سيأخذ الجزء الصحيح أي 6 وذلك بسبب وجود الجملة خارج أل {} لم يأخذ x=4 فلذلك تم استدعاء قيمة x البعيدة وليس القريبة أما عند استدعاء الدالة للمرة الثانية سيكون الناتج 7 وذلك لان القيمة المرسلة للدالة 6.5 وبما أن نوع البيانات للمتغير int x سيأخذ فقط الجزء الصحيح أي 6 ثم يزيد على قيمته بمقدار واحد فتصبح 7

11) ما ناتج تنفيذ البرنامج التالي:

`#include<iostream.h>` `int z=6;` `int f(int z){` `return ++z;}` `void g(int m){` `cout<<f(m)<<endl;}` `void main(){` `g(f(f(z)));` `cout<<z<<endl;}` ——— 9 6	في جملة الاستدعاء الأولى سيكون الناتج 9 لان الاستدعاء سيكون من الداخل للخارج أي في البداية سنرسل قيمة z التي تساوي 6 للدالة f فتقوم الدالة بزيادة قيمة z بمقدار واحد وتصبح قيمتها الجديدة 7 فيتم إرسال القيمة الجديدة للمتغير z للدالة f مرة أخرى وستقوم الدالة مرة أخرى بزيادة قيمة المتغير z بمقدار واحد وتصبح قيمتها الجديدة 8 فترسل قيمة المتغير الجديدة للدالة g التي تقوم بطباعة ناتج استدعاء الدالة f الذي يعمل على زيادة z بمقدار واحد وتصبح قيمته 9 ثم يقوم بطباعتها وبعدها عندما ننفذ جملة الطباعة سيطبع قيمة المتغير z الذي يساوي 6 لان التغيير الذي حصل عليه لا يخزن لان الاستدعاء (by value)

12) ما ناتج تنفيذ البرنامج التالي:

`#include<iostream.h>` `int fn(int w){` `if(w%2==0)` `return 1;` `else` `return w+fn(w-1);}` `void main(){` `int sum=0;` `for(int i=1;i<=4;i++)` `sum+=fn(i);` `cout<<sum<<endl;}` ——— 8	عند تنفيذ هذا البرنامج سيكون الناتج 8 فعند دخول جملة الدوران سيتم إرسال قيمة العداد i للدالة الذي تكون قيمته في البداية 1 فتقوم الدالة باستقبال القيمة من خلال المتغير w وتقوم من خلال جملة الشرط if التي تقوم بفحص هل العدد زوجي أم فردي فإذا كان زوجي يرجع واحد وإذا كان فردي سوف يجمع قيمة العدد مع ناتج استدعاء الدالة مرة أخرى من نفس الدالة ولكن لقيمة z-1 والفحص مرة أخرى وهكذا إلى أن يصل العداد إلى 5 فيقوم بالخروج من الدوران وطباعة ناتج جمع جميع القيم التي قامت بإرجاعها الدالة والتي ستكون 8

13) ما ناتج تنفيذ البرنامج التالي:

```
#include<iostream.h>
int as(int a){
if(a>=0)
return 1;
else
for(int i=1;i<3;i++)
return i+as(a+2);}
void main(){
int x=2;
cout<<as(x)<<endl;
x=-2;
cout<<as(x)<<endl;}
```

```
1
2
```

عند تنفيذ جملة الطباعة الأولى سيكون الناتج 1 وذلك لأنه سوف يرسل للدالة القيمة 2 وسيتم فحص القيمة 2 في الدالة فإذا كانت اكبر من 0 سوف يرجع واحد وعند تنفيذ جملة الطباعة الثانية سيكون الناتج 2 وذلك لان القيمة التي سترسل للدالة هي 2- وسيتم الفحص داخل الدالة إذا كانت اكبر من 0 وسيكون ناتج جملة الشرط if (false) وعندها سيدخل إلى else وسيدخل إلى جملة الدوران التي تكرر تنفيذ جملة return مرتين ولاكن ستأخذ ناتج تنفيذها في المرة الأولى فقط هي عندما تكون قيمة العداد i تساوي 1 وعندها سيتم جمع قيمته إلى ناتج استدعاء الدالة as وإرسال قيمة 2+2- التي تساوي 0 التي سوف تنتج 1 بعد إدخالها إلى جملة الشرط فيتم جمع القيمتين ويكون الناتج 2

14) اكتب برنامج يقوم بإيجاد مضروب عدد صحيح يتم إدخاله من لوحة المفاتيح باستخدام(recursive function)

```
#include<iostream.h>
int f(int);
void main(){
int x;
cin>>x;
cout<<f(x)<<endl;}
int f(int y){
if(y==1 || y==0)
return 1;
else
return y*f(y-1);}
```

قمنا بكتابة prototype للدالة f قبل الmain ثم قمنا بإدخال قيمة صحيحة من لوحة المفاتيح وهو العدد الذي سنقوم بحساب المضروب له ثم قمنا باستدعاء هذه الدالة عن طريق جملة الطباعة لكي يتم طباعة الناتج مباشرة وعرفنا الدالة بعد أل main بحيث أنها تستقبل العدد الذي سنحسب له المضروب عن طريق المتغير y ثم تقوم بفحص قيمة y هل هي تساوي 0 أو 1 فإذا كان نعم سيرجع 1 وإذا كان لا سيقوم بضرب العدد بناتج استدعاء الدالة نفسها ولكن(للعدد- 1) وسيبقى يستدعيه إلى أن تصبح قيمة المتغير 0 أو 1 ثم يرجع الناتج النهائي

15) اكتب برنامج يقوم بتعريف مصفوفة من بعد واحد حجمها 3 ثم قم باستدعاء دالة لتقوم بقراءة قيم وتخزينها في المصفوفة ثم استدعاء دالة أخرى للبحث عن وجود قيمة مدخلة من لوحة المفاتيح في المصفوفة.

قمنا بكتابة prototype للدالة read وكذلك للدالة search ثم قمنا بتعريف مصفوفة من بعد واحد حجمها 3 من نوع int ثم قمنا باستدعاء الدالة التي تقوم بإدخال القيم للمصفوفة (read) ثم قمنا بإدخال قيمة من لوحة المفاتيح ليتم البحث عنها في المصفوفة من خلال الدالة search التي تستقبل القيمة التي نريد أن نبحث عنها وتستقبل المصفوفة التي نريد البحث فيها وبعد أل main تم تعريف دالة إدخال القيم للمصفوفة read باستخدام جملة الدوران(كما ذكرنا سابقا) ثم عرفنا الدالة التي تقوم بالبحث وذلك أيضا باستخدام جملة دوران بحيث تقارن القيمة التي نريد أن نبحث عنها في المصفوفة مع قيم المصفوفة فإذا وجدت سيتم طباعة كلمة Found	`#include<iostream.h>` `void read(int []);` `void search(int,int []);` `void main(){` `int arr[3],v;` `read(arr);` `cin>>v;` `search(v,arr);}` `void read(int a[]){` `for(int i=0;i<3;i++)` `cin>>a[i];}` `void search(int x,int a[]){` `for(int i=0;i<3;i++)` `if(x==a[i])` `cout<<"Found"<<endl;}`

16) اكتب برنامج يقوم بتعريف مصفوفة من بعدين حجمها 3*3 ثم يقوم باستدعاء دالة تقوم بقراءة قيم المصفوفة ثم يتم استدعاء دالة أخرى تقوم بطباعة عناصر المصفوفة على شكل صفوف وأعمدة

قمنا بتخزين عدد الصفوف في الثابت row وكذلك عدد الأعمدة في الثابت col وذلك قبل ألmain (global) وقمنا بتعريف هذه الثوابت قبل ألmain لنستطيع استخدام هذه القيم في كل مكان ثم قمنا بكتابة prototype للدالة read وقمنا بتحديد عدد الأعمدة فيها (يجب فعل ذلك ويجوز لك أن تحدد عدد الصفوف أيضا) ثم قمنا بكتابة prototype للدالة print التي تقوم بطباعة قيم المصفوفة بشكل صفوف وأعمدة وعرفنا في أل main مصفوفة اسمها a وحددنا عدد صفوفها وأعمدتها من خلال الثوابت التي عرفناها في البداية ثم قمنا باستدعاء دالة إدخال القيم إلى المصفوفة وبعدها استدعينا الدالة التي تقوم بالطباعة print وبعد أل main قمنا بتعريف الدالة الأولى read لقراءة القيم وذلك باستخدام جملة دوران متداخلة(انتبه إلى عدد الصفوف والأعمدة لم نقوم بكتابتها أعداد ولكن استخدمنا الثابت لكي يعبر عن عدد الصفوف وكذلك الأعمدة) ثم عرفنا الدالة التي تقوم بطباعة قيم المصفوفة(كما تعلمنا سابقا).	`#include<iostream.h>` `const int row=3;` `const int col=3;` `void read(int[][col]);` `void print(int[][col]);` `void main(){` `int a[row][col];` `read(a);` `print(a);}` `void read(int b[][col]){` `for(int i=0;i<row;i++)` `for(int j=0;j<col;j++)` `cin>>b[i][j];}` `void print(int b[][col]){` `for(int i=0;i<row;i++)` `{for(int j=0;j<col;j++)` `cout<<b[i][j]<<" ";` `cout<<endl;}}`

17) اكتب برنامج يقوم بإرجاع قيمة أي عدد صحيح مرفوع الأس وذلك بعد إدخال العدد والأس باستخدام (recursive function)

قمنا بتعريف الدالة قبل أل main الذي يستقبل قيمتين من خلال المتغيرين base و exp حيث أن المتغير base سوف يخزن به العدد(الأساس) والمتغير exp سوف يخزن به الأس ثم نقوم بمقارنة الأس هل يساوي 1 فإذا كان نعم ارجع العدد نفسه(لان أي عدد مرفوع الأس واحد يكون العدد نفسه) وذا كان لا يساوي 1 نقوم بضرب العدد مع ناتج استدعاء الدالة نفسها ولكن بعد إرسال لها العدد و (الأس-1) وهكذا إلى أن يصل الأس إلى ال1 فيقوم بإرجاع الناتج النهائي. وفي أل main تم تعريف متغيرين لتخزين العدد في المتغير b والأس ليخزن في المتغير e ثم قمنا باستدعاء الدالة عن طريق جملة الطباعة بعد إرسال القيم لها	#include<iostream.h> int po(int base,int exp){ if(exp==1) return base; else return base*po(base,exp-1);} void main(){ int b,e; cin>>b>>e; cout<<po(b,e)<<endl;}

18) اكتب برنامج يقوم بتعريف مصفوفتين حجم كل واحدة 4 الأولى من نوع int والثانية من نوع char ثم يقوم باستدعاء دالتين الأولى للإدخال القيم إلى المصفوفة والأخرى تقوم بطباعة قيم المصفوفة.

قمنا بتعريف ثابت اسمه size وخزنا به حجم المصفوفتين 4 ثم استخدمنا template function لان لدينا مصفوفتين وكل مصفوفة لها نوع بيانات مختلف فاستخدمنا هذه الطريقة لتوفير الوقت والجهد بدلا من كتابة دالتين لكل مصفوفة ثم عرفنا دالة أخرى لطباعة قيم المصفوفة وأيضا سبقناها ب template function لنفس السبب ثم قمنا بتعريف مصفوفتين في أل main الأولى اسمها array1 حجمها size أي 4 والثانية اسمها array2 حجمها size أي 4 ثم قمنا باستدعاء الدالة read مرتين الأولى للمصفوفة الأولى والثانية للمصفوفة الثانية ثم قمنا باستدعاء دالة print مرتين أيضا الأولى للمصفوفة الأولى والثانية للمصفوفة الثانية.	#include<iostream.h> const int size=4; template<class t> void read(t a[]){ for(int i=0;i<size;i++) cin>>a[i];} template<class t> void print(t a[]){ for(int i=0;i<size;i++) cout<<a[i]<<endl;} void main(){ int array1[size]; char array2[size]; read(array1); read(array2); print(array1); print(array2);}

19) اكتب برنامج يقوم بتعريف مصفوفتين حجم كل واحدة 3*3 ثم يقوم بإدخال القيم للمصفوفة الأولى ثم يقوم بنسخ قيم المصفوفة الأولى إلى المصفوفة الثانية ثم يقوم بطباعة قيم المصفوفة الثانية بعد نسخ القيم إليها.

`#include<iostream.h>` `const int row=3;` `const int col=3;` `void read(int a[][col]){` `for(int i=0;i<row;i++)` `for(int j=0;j<col;j++)` `cin>>a[i][j];}` `void copy(int a[][col],int b[][col]){` `for(int i=0;i<row;i++)` `for(int j=0;j<col;j++)` `b[i][j]=a[i][j];}` `void print(int a[][col]){` `for(int i=0;i<row;i++)` `{for(int j=0;j<col;j++)` `cout<<a[i][j]<<" ";` `cout<<endl;}}` `void main(){` `int a[row][col],b[row][col];` `read(a);` `copy(a,b);` `print(b);}`	عرفنا ثابتين الأول لنخزن به عدد الصفوف والثاني لنخزن به عدد الأعمدة وقمنا بتعريف دالة لقراءة القيم وإدخالها إلى المصفوفة الأولى a ثم قمنا بتعريف دالة أخرى اسمها copy ونلاحظ بأنها تستقبل مصفوفتين لكي تقوم بنسخ القيم الموجودة في المصفوفة الأولى a إلى المصفوفة الثانية b وذلك باستخدام جملة الدوران المتداخلة ثم نسخ كل القيم من المصفوفة الأولى إلى المصفوفة الثانية ثم قمنا بتعريف دالة أخرى تقوم بطباعة قيم المصفوفة على شكل صفوف وأعمدة ثم قمنا بتعريف مصفوفتين a و b في أل main وكلا المصفوفتين لها الحجم 3*3 ثم قمنا باستدعاء الدالة read وأرسلنا إليها المصفوفة الأولى a لكي يتم تخزين القيم في المصفوفة الأولى ثم قمنا باستدعاء دالة copy وأرسلنا إليها مصفوفتين a و b لكي يقوم بنسخ القيم من المصفوفة الأولى التي قمنا بإدخال القيم إليها من خلال الدالة read إلى المصفوفة الثانية ثم قمنا باستدعاء دالة print لكي تقوم بطباعة قيم المصفوفة الثانية b لأننا أرسلنا لهذه الدالة المصفوفة b.

20) اكتب برنامج يقوم بتعريف دالة تقوم بقراءة حرف من لوحة المفاتيح وتعريف دالة أخرى تقوم بإرجاع الحرف الذي يليه

`#include<iostream.h>` `char c;` `void c1(){` `cin>>c;}` `char c2(int c){` `return ++c;}` `void main(){` `c1();` `cout<<c2(c)<<endl;}`	قمنا بتعريف متغير عالمي global اسمه c لكي تستطيع جميع الدوال استخدامه ثم قمنا بتعريف دالة اسمها c1 تقوم بإدخال حرف من لوحة المفاتيح وتخزينه في المتغير c ثم قمنا بتعريف دالة أخرى اسمها c2 ترجع حرف وتستقبل حرف تستقبل الحرف المدخل وترجع الحرف الذي يلي الحرف المدخل تماما وذلك بزيادة قيمة المتغير c بمقدار واحد ثم قمنا في أل main باستدعاء الدالة c1 بشكل منفرد لنقوم بإدخال حرف ثم قمنا باستدعاء الدالة c2 عن طريق جملة الطباعة لطباعة الحرف التالي مباشرة على الشاشة.

المؤشرات (Pointers)

-- المؤشر: هو عبارة عن متغير يحتوي على عنوان للمتغير الذي يؤشر عليه في الذاكرة ويكون هذا العنوان قيمة لهذا المؤشر.

- أنا متأكد بأنك تسأل نفسك بماذا يختلف المؤشر عن المتغير العادي ؟؟؟
أجيبك يا صديقي أن المؤشر قيمته هي عنوان المتغير الذي يؤشر عليه هذا المؤشر والاختلاف الأخر هو أن المتغير يعتبر المرجع المباشر للقيمة المخزنة فيه أما المؤشر يعتبر المرجع غير المباشر للقيمة المخزنة في المتغير الذي يؤشر عليه.

- هل اعرف المؤشر كما عرفنا المتغيرات العادية ؟؟؟
نعم,ولكن ضع قبل اسم المتغير إشارة نجمة (*)

إليك هذه الأمثلة على تعريف المؤشرات :

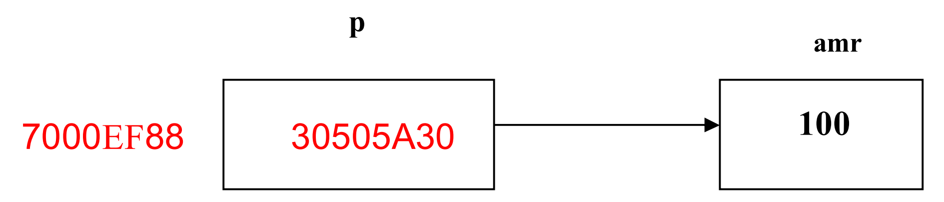
int *x; : هنا عرفنا مؤشر اسمه x ويؤشر على قيمة صحيحة أي (int)
Char *ptr; : عرفنا مؤشر اسمه ptr ويؤشر على قيمة حرفية أي (char)

- المهم الآن أن تعرف كيف تعطي المؤشر عنوان المتغير:
سوف نستخدم لذلك ما يسمى معامل العنوان (&) , إليك هذا المثال لعله يساعدك في الفهم :

```
int amr=100;
int *p;
p=&amr;
```

-- في الجملة الأولى عرفنا متغير اسمه amr وأعطيناه القيمة 100 وفي الجملة الثانية تم تعريف مؤشر باسم p أما في الجملة الثالثة تم إعطاء المؤشر قيمة , ما هي هذه القيمة؟ إنها عنوان المتغير amr وذلك باستخدام (&) أي هنا أصبح المؤشر هو مرجع لقيمة المتغير amr التي هي 100 ولكن بشكل غير مباشر (أي أستطيع أن احصل على القيمة 100 من خلا المؤشر p) .

** إليك هذا الرسم للتوضيح :-
افرض انه تم حجز الموقع 30505A30 للمتغير amr وحجز الموقع 7000EF88 للمؤشر p



ما رأيك إذا قمت بتنفيذ الجملتين التاليتين, فماذا تتوقع الناتج ؟؟؟

```
cout<<*p;
cout<<amr;
```

لا تستغرب إذا قلت لك أن الناتج سيكون نفسه وطباعة القيمة 100 وذلك لان اسم المؤشر سبق ب (*) التي تعني القيمة الموجودة في الموقع الذي يؤشر عليه هذا المؤشر.

أما لو نفذنا الجملة التالية:

cout<<p;

سيقوم بطباعة 30505A30 وهو عنوان المتغير amr التي هي قيمة المؤشر p.

ولو نفذنا الجملة هذه:

***p=60;**

سوف تصبح قيمة المتغير amr الجديدة هي 60 وذلك لان المؤشر عبارة عن مرجع غير مباشر للقيمة الموجودة في المتغير.

وعند تنفيذ الجملة التالية:

cout<<&amr;

سوف يتم طباعة موقع المتغير amr ويطبع القيمة 30505A30 .

أما عند تنفيذ الجملة التالية:

cout<<*&p;

سوف يطبع 30505A30 وهي قيمة المؤشر p .

وكذلك لو نفذنا هذه الجملة:

cout<<&*p;

سوف يطبع أيضا قيمة المؤشر وهي 30505A30 .

-- أعطيك ملخص لما سبقت ذكره:

1) عند طباعة p تعطي نفس ناتج طباعة &amr وهذا يكون عنوان المتغير amr .
2) عند طباعة *p تعطي نفس ناتج طباعة amr وهذا يكون قيمة المتغير amr .
3) عند طباعة &*p أو *&p سيطبع عنوان المتغير amr وهو الموجود في المؤشر (فلذلك نستنتج أن طباعة *&P أو &*p نفس طباعة p أي أن الإشارتين * و & تلغيان بعضهما البعض عند استخدامهما معا).

فلو قمنا بتعريف مؤشر أخر اسمه pp كالتالي:

int *pp;

ونفذنا الجملة التالية:

pp=p;

هذا يعني أن القيمة الموجودة في المؤشر p أصبحت موجودة أيضا في المؤشر pp وكما نعلم أن القيمة الموجودة في المؤشر p هي عنوان المتغير amr وهي 30505A30 أي أصبح المؤشر pp هو أيضا مؤشر على المتغير amr وأصبح مرجع غير مباشر للقيمة الموجودة فيه.

ولو نفذنا الجملة التالية:

cout<<*pp;

سيقوم بطباعة القيمة 60 .
وكل الجملة التي نفذناها على المؤشر p ستعطي نفس النتائج إذا طبقت على المؤشر الجديد pp .

*** التعابير الرياضية مع المؤشرات:-

سأعطي لك اولويات المعاملات الرياضية واولويات معاملات المؤشرات (* , &) :-

1- الأقواس (,)
2- معامل المؤشر (*) ومعامل العنوان (&) ومعامل الزيادة (++) ومعامل النقصان (- -)
3- القسمة (/) وباقي القسمة (%) والضرب (*)
4- الجمع (+) والطرح (-)

- ملاحظة مهمة:- لا تنسى إذا تساوت الأولوية يصبح التنفيذ من اليسار لليمين إلا في الحالة رقم (2) يتم التنفيذ من اليمين إلى اليسار.

إليك هذا المثال :-

```
int n,k,*ptr;
n=7;
ptr=&n;
k=*ptr+1;
```

- نلاحظ من المثال السابق أن الجملة الأخيرة تكافىء:-

```
k=n+1;
```

ولكن ماذا لو أردنا زيادة قيمة المتغير n بواسطة معامل الزيادة (++) ؟؟؟؟
أنا متأكد انك تقول أن الجملة المناسبة هي:

```
*ptr++;
```

ولكن أنا أقول لك انك قد نسيت أن الأولوية هنا متساوية بين معامل المؤشرات (*) ومعامل الزيادة (++) وستنفذ هذه الجملة من اليمين لليسار ((تذكر الملاحظة المهمة التي ذكرتها لك))

فتصبح هذه الجملة عبارة عن هاتان الجملتان:-

```
ptr++;
*ptr;
```

وهذا سوف يزيد القيمة الموجودة في المؤشر ptr والتي هي عنوان المتغير n فمثلا لو كان عنوان المتغير n هو 11499001 فبعد تنفيذ الجملة ستصبح قيمة المؤشر ptr هي 11499005 وذلك لان نوع البيانات هو int وكما نعلم أن نوع البيانات int يأخذ (4 بايت) ويرجع قيمة غير متوقعة ولكن ليس هذا الذي نريده نحن نريد أن نزيد قيمة المتغير n بواسطة المؤشر الذي يؤشر عليها (ptr)

ولكي نتغلب على هذه المشكلة سوف نستخدم الأقواس لجعل الأولوية للمعامل (*) كالتالي:

```
(*ptr)++;
```

حيث انه سوف يقوم بإحضار القيمة الموجودة في الموقع الذي يؤشر عليه المؤشر ptr ثم زيادة هذه القيمة بمقدار واحد.

ونسطع أيضا أن نحل المشكلة بكتابة الجملة كالتالي:

```
++*ptr;
```

حيث هنا الأولوية ستكون لمعامل المؤشرات (*) ثم لمعامل الزيادة (++) .

- ملاحظة:- ما ينطبق على معامل الزيادة (++) ينطبق على معامل النقصان (- -).
-

■ هذه مجموعة من الأجزاء البرمجية مع النتائج المتوقعة بعد تنفيذها لعلها تزيد مهارتك ومعرفتك:

[1]

```
int x=0,y=0;
x=2;  y=4;
x=x+x;  x=x+y;
x++;
y=x;  x=y;
y++;
cout<<++x;
cout<<x++<<endl;
cout<<y++<<endl;
cout<<x;
cout<<y<<endl;
```

Output [1]

```
1010
10
1111
```

[2]

```
for(int i=0;i<7;i+=0.6)
cout<<"Cout<<"<<i++<<endl;
cout<<i<<i<<endl;
```

Output [2]

```
Cout<<0
Cout<<1
Cout<<2
Cout<<3
Cout<<4
Cout<<5
Cout<<6
77
```

[3]

```
for(int i=2;i<=8;i++)
if((i%2)==0)
cout<<"Amr"<<endl;
```

Output [3]

```
Amr
Amr
Amr
Amr
```

[4]

```
for(int i=1;i<=3;i+=2)
for(int j=2;j<=i+1;j+=3)
{for(int k=4;k>2;k-=3)
cout<<i<<" "<<j<<" "<<k<<endl;
cout<<i<<" "<<j<<" "<<k<<endl;}
```

Output [4]

```
1 2 4
1 2 1
3 2 4
3 2 1
```

[5]

```
int a=2,b=10;
cout<<"a\t\n";
cout<<"\\\\\r\"";
cout<<"BBB\r\"\B\"";
cout<<a<<"\r"<<b<<endl;
cout<<b<<"\r"<<"\'\'"<<"\n";
```

Output [5]

```
a
10"2
' '
```

[6]

```
char a[4]={'a','b','c','d'};
char *p1;  int n=2,*p2;
p1=a;
p2=&n;
cout<<p1[3]<<endl;
cout<<p1[++*p2]<<endl;
```

Output [6]

```
d
d
```

[7]

```
int x=2.5;
while(x<=3.2)
{
cout<<floor(x);
cout<<x-floor(x)<<'\n';
x+=1;
}
```

Output [7]

```
20
30
```

[8]

```
int xx=2,yy=4;
while((xx+yy)%xx==0)
{
if(xx+yy<=20)
{xx+=4; yy+=2;}
cout<<xx<<" \t"<<yy<<"\n";
}
```

Output [8]

```
 6      6
10      8
```

[9]

```
void vvv1()
{int x=2;}
void vvv2()
{int x=4; vvv1(); }
void vvv3(int x)
{cout<<++x<<endl; }
void vvv4(int &x)
{ x+=3; }
void vvv()
{
int x=2; vvv1();  vvv2();
while(x<5)
{vvv1();  vvv3(x);  vvv4(x);}
}
void main()
{int x=7; vvv();  cout<<x<<endl; }
```

Output [9]

```
3
7
```

[10]

```
int intt1(int &int1)
{ int1=int1+int1; ++int1; return int1; }
int intt2(int int1,int &int2)
{
int1=int1+int2;
int2=int1+int1;
return int2;
}
void main( )
{
int int1,int2;
int1=10; int2=10;
int int3=intt1(int1);
int int4=intt2(int1,int2);
cout<<int1<<endl;
cout<<int2<<endl;
cout<<int3<<endl;
cout<<"int int;\nint int;\nint int;\n";
}
```

Output [10]

```
21
62
21
int int;
int int;
in tint;
```

[11]

```
int amr[]={65,77,82};
char cs[3];
cout<<"Directed By:";
for(int crt=0;crt<3;crt++)
{ cs[crt]=amr[crt];
  cout<<cs[crt]; }
```

Output [11]

Directed By:AMR

[12]

```
int y1=2,y2=0.2,y3=3.2;
int x1=pow(y1,sqrt(16));
int x2=fmod(y3,y2);
int x3=pow(ceil(y3),ceil(y2)+1);
int x4=-x1;
cout<<fabs(x1)<<endl;
cout<<fabs(x2)<<endl;
cout<<fabs(x3)<<endl;
cout<<fabs(x4)<<endl;
```

Output [12]

16
0
3
16

[13]

```
template<class t>
t TF1(t x,t y,t z);
template<class s>
s TF2(s y1,s &y2);
void main( )
{
int x=10,y=10;
cout<<TF1(++x,x++,x)<<endl;
cout<<TF2(y++,++y)<<endl;
}
template<class t>
t TF1(t x,t y,t z)
{t a=x-y+z;  return a; }
template<class s>
s TF2(s y1,s &y2)
{ s tt=++y2+y1;  return tt; }
```

Output [13]

11
24

[14]

```
int over(int m1)
{ return m1+5+m1++; }
double over(float f)
{ return f++; }
double over(int a,int b)
{ return a*b; }
double over(double a,double b)
{ return a*b*a; }
void main( )
{
for(int i=1;i<=6;i++)
{   cout<<over(i)<<endl; i=i+1;
    cout<<over(i)<<endl; ++i;
    cout<<over(i++,i++)<<endl;  i=i+1;
    cout<<over(i+=1,i)<<endl;
}
}
```

Output [14]

7
9
9
42

[15]

```
char x='A',y='Z',z;
char *c; c=&x;
cout<<++*c;
z=*c+=3;
c++;   c=&y;
cout<<--*c;  cout<<z;
```

Output [15]

BYE

[16]

```
int y=2;
void main( )
{  int y=3; cout<<y<<endl;
   cout<<::y<<endl<<y<<endl;
   cout<<y+::y<<endl;
}
```

Output [16]

3
2
3
5

[17]

```
int s=10;
void ss(int &n)
{ cout<<++n<<endl; }
void main( )
{
int s=5; ss(s);  ss(::s);
}
```

Output [17]

6
11

[18]

```
double duoble(static double x)
{ x++; x+=2; return x;  }
void main( )
{
for(int i=2;i<=4;i++)
if(i<=18)
cout<<duoble(i)<<endl;
```

Output [18]

5
6
7

[19]

```
int x1=7.3; int x2=8.9;
void sf(int x1,int x2)
{++x1;x2++;cout<<x1+(++x1)<<'\n';}
void main( )
{
int x2=2;
{int x1=5; sf(x1,x2); }
sf(x1,x2);
}
```

Output [19]

14
18

[20]

```
int i=2;
while(++i<5)
{ cout<<i<<endl;
  if(i%2==0)
  if(i>=4)
  { continue; i=i+2; }
   cout<<i<<endl; }
```

Output [20]

3
3
4

[21]

```
int w=6;
void p2(int,int);
void p1(int w)
{ for(int i=2;i<=2;i++)
    p2(i,w);
}
void p2(int w,int i)
{  int c=0;
   cout<<w<<" "<<c<<" "<<i<<endl;
   cout<<w++<<c<<++w<<endl;
}
void main( )
{ p1(w); }
```

Output [21]

```
2 0 6
303
```

[22]

```
int in=8;
int v(int [],int);
void main( )
{   int arr[5];
     for(int in=0;in<::in;in++)
     cout<<v(arr,in)<<endl;}
int v(int arr[],int n)
{
        in=5;
        return arr[n]=n;
}
```

Output [22]

```
0
1
2
3
4
```

[23]

```
void inc(int &i)
{ cout<<++i<<endl; }
int out(int i)
{ return pow(i,2); }
void main( )
      { double x=2.5;
       for(int i=2;i<=ceil(x);inc(i))
      { cout<<out(i)<<endl; }}
```

Output [23]

```
4
3
9
4
```

[24]

```
void f1(int);
int f2()
{ int i=pow(floor(0.9),ceil(100));
  return i; }
void f3(int n)
{ n=n+3;}
void f1(int n)
{ if(n<=9)
cout<<f2()<<"\n";  f3(n);
cout<<"END"<<endl;}
void main()
{ int n=2;  f1(n); }
```

Output [24]

```
0
END
```

الآن ما رأيك بان تفحص نفسك وتجيب على هذا الامتحان البسيط لتقيس قدرتك؟؟؟ →

[General Exam]

Q1) Choose the correct answer:

1) One Of the following statements its output is 66 :
a) cout<<"6\r6"; b) cout<<"6"+"6";
c) cout<<"6\n6"; d) cout<<"666\r66 ";

2) One of the following statements it's not correct:
a) cout<<"abc/rbb"; b) cout<<"a<<endl;";
c) cout<<"cout<<\"10\""; c) cout<<"\\\\\";

3) After execute this statement what is the output:
cout<<(true?"True":"False");
a) True b) False c) No Output d) there is syntax error

4) The variables which passes to the function at invoke time called:
a) parameters b) arguments c)constants d)a+b

5) The error in this IF_Statement consider
If(5<2);
Cout<<"if(if(if))"<<endl;
a) Syntax error b) logical error c) runtime error d) a+b

Q2) Write the following statements in just one equivalent statement:

1	Cout<<"ABC\r"; Cout<<"\n"; Cout<<endl<<"ABC\r";	→	
2	if(x>3) cout<<++x; else cout<<x++;	→	
3	int x; int y; x=3;	→	

Q3) When I wrote these codes in c++ I was so tired and confused so that may be there are syntax errors, you must discover these errors if they are found and correct them

<table>
<tr><th>1</th><th>2</th></tr>
<tr><td><pre>int x=6.2;
++6.2;
for(int i=0:i<10:)
{ cout>>"I:">>i>>endl;
  i++;
}
y=2;
cout<<y<<endl;</pre></td><td><pre>Int x=5;
switch(x>2)
{ case true
  cout<<"max"
  case false
  cout<<"min"
}</pre></td></tr>
<tr><td></td><td></td></tr>
</table>

Q4) Do the following tasks:

1) Define an 2-Dimensional array and give it initial values as this table.

-1	3	22
5	-4	19
0	10	87

2) Define function to calculate the summation of even numbers between 2 and 12 by using recursive function.
3) Check the value of variable x if it one of the following letters (A,B,C,D)
4) Define the template function takes two parameters and return the summation for own parameters
5) Print the following form by using For-Statement

```
#
#
####
####
#########
#########
```

Q5) What is the output for these sections codes:

[1]

```
Int i=2;
Do{
++i;
i++;
i=(++i)+(i++);
cout<<i<<endl;
}
while(i++>=10);
```

Output [1]

[2]

```
Int j=1;
Do{
int i=1;
Do{
cout<<i++<<endl;
I++;
}
while(i<5);
Cout<<j++;
}
While(j<3);
```

Output [2]

[3]

```
Void print(int &x)
{ if(x>=10);
  retutn x++;
}
void main( )
{  int y=7;
   cout<<print(2)<<endl;
   cout<<print(&y)<<endl;
   cout<<print(y)<<endl; }
```

Output [3]

[4]

```
Static int x1=10; int x2=5;
Void N(int x1,int x2)
{ x1=100;  cout<<x1++<<++x2;  }
void main( )
{
int x2=10;
{ N(x1,x2); }
}
```

Output [4]

[5]

```
Cout<<"\\\\\\\n";
Cout<<(10>0?"\"":"\"");
Cout<<"/\r\\\\"<<endl;
```

Output [5]

[6]

```
Int *p1,*p2;  int x=0;
char c={'A','M','R'};
Cout<<c[x]++<<c[++x]<<c[x++]--;
P1=&x;  p2=c; *p1=2;
Cout<<++(*p)<<*(p2++)<<endl;
```

Output [6]

Q6) Write a full programs in c++ language to:

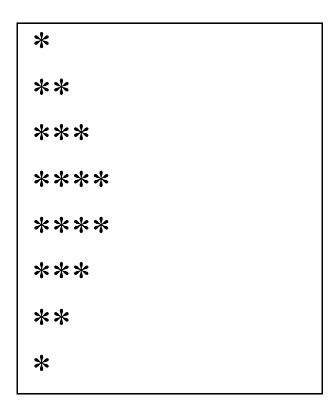
1) Find the value of variable S by using Loop-Statement
S=3/4+5/7+7/8+9/11+11/12

2) Insert 10 values into 1-Dimensional array then find the summation of numbers in this array that divided by 5

3) Insert real number then print the integer and fractional parts

4) Insert 8 values into 2-Dimensional array then fined a number of positive numbers and negative numbers in this array

5) Print the following stars form by using loop statements



The End